VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

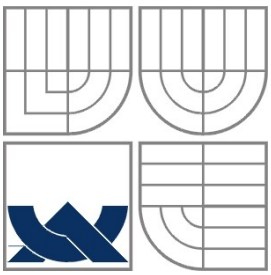
FLEXIBILNÍ MĚŘENÍ TOKŮ NA SÍTI

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

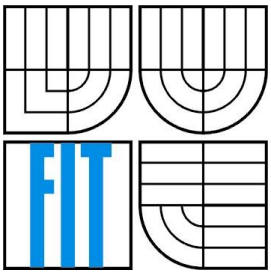
AUTOR PRÁCE
AUTHOR

Bc. Ladislav Varga

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

FLEXIBILNÍ MĚŘENÍ TOKŮ NA SÍTI

FLEXIBLE NETWORK FLOW MEASUREMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Ladislav Varga
AUTHOR

VEDOUCÍ PRÁCE — Ing. Martin Žádník
SUPERVISOR

BRNO 2010

Abstrakt

Táto diplomová práca sa zaoberá návrhom a implementáciou sondy pre meranie tokov na sieti. Obsahuje teoretický rozbor problematiky merania, popis algoritmov a techník používaných pri meraní na báze tokov. Pri návrhu architektúry sondy je kladený dôraz na efektívnu indexáciu záznamov tokov a flexibilitu záznamu, tak aby bola užívateľovi umožnená parametrizácia merania.

Abstract

This thesis deals with designing the probe used for measuring network flows. It contains theoretical analysis of network measurment topic, description of algorithms and principles used for network flow measurement. Emphasis on the probe architecture lies on efficient indexing algorithm and flow record flexibility, such that user is able to define format of flow record.

Klíčová slova

tok, sonda, IPFIX, meranie sietí, meranie tokov, flexibilné meranie tokov

Keywords

flow, probe, IPFIX, network measurement, flow measurement, flexible flow measurement

Citace

Ladislav Varga: Flexibilní měření toků na síti, diplomová práce, Brno, FIT VUT v Brně, 2010

Flexibilní měření toků na síti

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením Ing. Martina Žádníka. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Ladislav Varga
20.5.2010

Poďakovanie

Touto cestou chcem poďakovať Ing. Martinovi Žádníkovi za jeho podnety a rady pre vypracovanie tejto diplomovej práce.

Obsah

1 Úvod

Rozvoj informačných technológií v druhej polovici 20. storočia so sebou priniesol vznik dátových sietí, ktorých používanie sa postupne rozšírilo z obranných zložiek do ostatných oblastí našej spoločnosti. Došlo ku komercializácií vývoja a používania sieťových technológií a vzájomným prepájaním jednotlivých sietí vznikla celosvetová sieť, ktorú dnes poznáme pod názvom Internet. Za posledných dvadsať rokov zaznamenal Internet neočakávaný rozmach a dnes je k nemu pripojených vyše 1,7 miliardy ľudí [8]. S rastúcim počtom užívateľov rastú nielen možnosti využitia, ale aj nároky na prenosové rýchlosti, bezpečnosť a komplexnosť Internetu.

Často je užitočné alebo potrebné mať k dispozícií informácie o stave siete a dátach tečúcich jednotlivými bodmi siete. Jednou z dôležitých činností pri prevádzkovaní Internetu je preto monitorovanie parametrov sietí a prenosov na nich. Poskytuje správcom jediný prehľad o aktuálnom dianí na sieti. Meranie a monitorovanie siete je potrebné pre širokú škálu aplikácií, medzi ktoré patrí účtovanie poplatkov v závislosti od objemu prenesených dát, traffic engineering používaný pre optimalizáciu vyťaženia sietí, plánovanie a návrh nových sietí alebo detekcia rôznych typov útokov (denial-of-service útoky, šírenie červov, skenovanie portov apod.).

Pre meranie na sieťach existuje niekoľko rôznych prístupov, ktorými sú aktívne meranie, pasívne meranie a meranie na báze tokov. Práve posledná menovaná metóda sa stala najviac populárnou v posledných rokoch a je aj predmetom tejto práce. Cieľom práce je naštudovať princípy a techniky používané pre meranie tokov na IP sieťach a navrhnúť architektúru sondy, ktorá bude schopná merať premávku na vysokorýchlostných sieťach. Druh a formát zaznamenávaných dát bude definovateľný užívateľom a zo sondy budú odosielané vo formáte IPFIX. Cieľovou platformou pre implementáciu sondy je operačný systém Linux.

V druhej kapitole práce sú popísané spôsoby merania v počítačových sieťach obecne a meranie založené na báze tokov, na ktoré je táto diplomová práca zameraná. Podrobnejšie sa venuje technikám a algoritmom používaným pri meraní tokov a popisu protokolu pre export nameraných dát – IPFIX. Tretia kapitola obsahuje návrh sondy pre meranie tokov. Popisuje architektúru sondy, použité princípy a algoritmy a taktiež spôsob konfigurácie formátu meraných dát. Štvrtá kapitola práce sa venuje implementácií sondy. Rozoberá motívy pre voľbu použitých implementačných nástrojov a štruktúru implementovanej aplikácie. Piata kapitola je zameraná na experimenty a meranie výkonnosti sondy. Záver práce rekapituluje dosiahnuté výsledky a možné optimalizácie.

2 Siete a ich meranie

Internet ako aj veľká časť privátnych sietí sú dnes implementované pomocou sady komunikačných protokolov, ktoré sú súčasťou tzv. Internet modelu. Tento model definuje logickú štruktúru sieťovej komunikácie a je rozdelený do 4 vrstiev – aplikačná, transportná, internetová a linková. Protokol, vo význame v akom budeme tento termín používať, je množina pravidiel, ktoré definujú formát a reprezentáciu dát a spôsob ich doručovania prostredníctvom komunikačných kanálov. Paket je dátová jednotka doručovaná po sieti, naformátovaná podľa pravidiel určitého protokolu. Protokoly operujú na jednotlivých vrstvách Internet modelu podľa toho, akú funkciu pri doručovaní dát zabezpečujú. Nás budú z pohľadu merania sietí primárne zaujímať protokoly internetovej a transportnej vrstvy, ktoré sa starajú o adresovanie, smerovanie paketov medzi zariadeniami a ich doručovanie správnemu procesu v rámci zariadenia. Z hlavičiek týchto paketov môžme získať informácie potrebné pre meranie a monitorovanie sieťovej prevádzky.

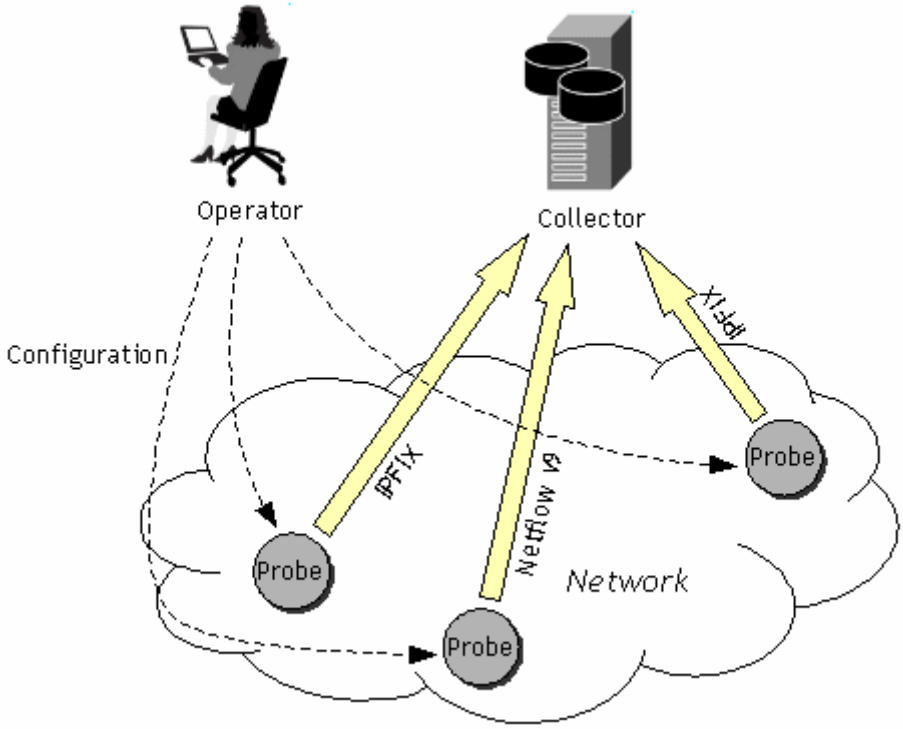
Meranie na sieti môžeme definovať ako periodickú činnosť zisťovania stavu uzlov siete a zbieranie informácií o komunikácií prebiehajúcej cez tieto uzly [9]. Tieto informácie sú potrebné pre zabezpečenie spoľahlivého chodu siete a konkrétne prípady ich použitia budú prebrané ďalej v tejto kapitole. Existujú 3 základné prístupy k meraniu sietí a jedným z nich je **aktívne meranie**. Pri aktívnom meraní sa snažíme získať parametre siete medzi koncovými zariadeniami. K tomuto účelu vysiela aktívne zariadenie do siete pakety a počas ich putovania sieťou získava potrebné informácie, ako napríklad uzly, ktorými paket prechádza, oneskorenie paketov, atp. Tento druh merania sa používa napríklad na mapovanie topológie siete, zisťovanie priepustnosti a oneskorenia na sieti. Neposkytuje však žiadne údaje o premávke práve prebiehajúcej na sieti. Typické nástroje používajúce tento druh merania sú `ping` a `traceroute`.

Druhým spôsobom merania je **pasívne meranie**. Pri pasívnom meraní negenerujeme do siete žiadne vlastné pakety, ale ukladáme tie, ktoré pretekajú daným ziariadením, prípadne štatistiky o nich. Ukladanie celých paketov je však kapacitne príliš náročné a pri dnešných objemoch dát pretekajúcich sieťami prakticky nemožné. Pre zbieranie štatistík o paketoch sa dnes často používa databáza MIB-II (Management Information Base) a protokol SNMP. Na aktívnom prvku siete beží SNMP agent, ktorý na vyžiadanie zasiela vzdialenému SNMP manažéru informácie z MIB databázy daného zariadenia. Štatistiky o paketoch sa dajú použiť pre sledovanie objemu prípadne typu premávky na sieti, strácame však informácie z obsahu paketu a nemáme tak možnosť sledovať presnejšie tok paketov na sieti.

Kompromisom medzi ukladaním celých paketov a štatistík o nich je **meranie založené na tokoch**. Pri tomto prístupe sú zhromažďované informácie o paketoch na základe určitých spoločných vlastností paketov. Každý meraný paket sa analyzuje a kľúčové informácie v ňom obsiahnuté sú použité na jeho priradenie ku konkrétnemu toku. V zázname daného toku sú následne zohľadnené parametre paketu, ktorý bol práve identifikovaný ako paket náležiaci tomuto toku. Tento prístup je dnes najčastejšie používaným pri meraní sieťovej premávky a je predmetom tejto práce. Podrobnejšie popísaný je v nasledujúcej podkapitole.

2.1 Meranie na báze tokov

Na dáta tečúce IP sieťou sa môžme pozerať ako na toky, ktoré klasifikujeme podľa určitých kritérií. **Tok** je obvykle definovaný ako jednosmerný prúd IP paketov, ktoré sú identifikované spoločnou päticou – typ protokolu, zdrojová IP adresa, cieľová IP adresa, zdrojový port, cieľový port. V tomto ponímaní môže jednému toku prislúchať až niekoľko tisíc paketov. Pakety sú priraďované tokom na základe vyhodnotenia vlastností každého paketu. Pakety so spoločnými vlastnosťami náležia rovnakému toku. **Záznam toku** obsahuje informácie o danom toku. Medzi ne patria namerané dáta (napr. celkový objem dát, počet paketov) a charakteristické vlastnosti toku, ktoré nazývame **kľúčové polia** (tj. zdrojová IP adresa, ...).



Meranie tokov môže vykonávať smerovač počas procesu smerovania paketov, samostatné zariadenie zapojené medzi dva aktívne prvky siete alebo sonda pripojená k linke prípadne k sieťovému rozhraniu. Architektúra používaná pri meraní tokov je zobrazená na obrázku 2.1.Obrázok 2.1: Meranie na báze tokov (inšpirované z [20]).

Sonda (probe) zabezpečuje získavanie paketov na monitorovanom mieste v sieti, ich priradenie príslušnému toku, ukladanie záznamov o aktívnych tokoch a odosielanie dát o ukončených tokoch kolektoru. **Kolektor** (collector) je zariadenie, ktoré zbiera a ukladá záznamy o nameraných tokoch. Môže ich spracovávať, analyzovať a ďalej poskytovať (napríklad pre zobrazenie). Namerané dáta sú

medzi sondami a kolektorom prenášané prostredníctvom niektorého z protokolov pre export dát. Najpoužívanejšími sú protokoly NetFlow v5 a NetFlow v9 od firmy Cisco a protokol IPFIX. Operátor má zvyčajne možnosť vzdialene sondy konfigurovať a tým definovať parametre merania.

Po získaní paketu musí sonda z jeho hlavičky extrahovať kľúčové polia, ktoré následne použije pre **vyhľadanie toku**, ku ktorému paket patrí. Po vyhľadaní toku nasleduje **aktualizácia záznamu**, pokiaľ už daný tok existuje, alebo vloženie nového záznamu ak záznam toku zatiaľ neexistuje. Pri aktualizácií záznamu toku pridáme štatistiky získané z paketu k štatistikám toku. Súčasne s týmto procesom musí bežať **proces údržby**. Ten kontroluje stav tokov, ktorých záznamy sú udržované v pamäti. Ak je potrebné niektorý z tokov expirovať, proces údržby musí tento stav rozpoznať a postarať sa o ukončenie prípadne export záznamu toku. Tok môže byť považovaný za ukončený z nasledovných dôvodov:

- po určitú dobu neprišiel žiadny paket, ktorý by patril danému toku – **neaktívny timeout**
- tok existuje príliš dlho – **aktívny timeout**
- obdržali sme paket s TCP príznakom FIN alebo RST, ktoré explicitne vyžadujú ukončenie spojenia na transportnej vrstve

Techniky, princípy a algoritmy používané pri jednotlivých procesoch merania sú popísané v ďalších častiach tejto kapitoly.

2.2 Zachytávanie paketov

Aby sme mohli spracovávať údaje o sieťovej premávke, potrebujeme mať prístup k paketom, ktoré ju tvoria. Zachytávať pakety pre ich ďalšie spracovanie je možné rôznymi prístupmi a na tento účel je v operačnom systéme Linux dostupných niekoľko hotových riešení, z ktorých niektoré sú popísané v tejto kapitole.

Keď sieťová karta obdrží ethernetový rámec, ktorého cieľová MAC adresa sa zhoduje s jej MAC adresou, vygeneruje požiadavku na prerušenie. Obslužnú rutinu tohto prerušenia obsahuje ovládač sieťovej karty, ktorý paket skopíruje z vyrovnávacej pamäte sieťovej karty do pamäti jadra operačného systému. Potom skontroluje typ prijatého paketu a odošle ho obslužnej rutine príslušného protokolu.

2.2.1 Ulogd

Pri spracovaní paketov v jadre operačného systému zvyčajne dochádza k ich filtrovaniu (technika používaná napr. k implementácií firewallov). ulogd je démon, ktorý prijíma pakety, poslané do cieľa ULOG v Linuxovom paketovom filtri netfilter/iptables.

Keď je paket poslaný do cieľa ULOG, kernel ho rozošle multicastom cez netlink soket. ulogd sa pri spustení prihlási do multicastovej skupiny a prijíma pakety, ktoré môžu byť na tomto mieste užívateľom ďalej spracované. ULOG môže byť nastavený tak, aby namiesto celých paketov iba kopíroval ich hlavičky a následne ich posielal vo väčších dávkach. Tým sa výrazne zníži réžia pri získavaní paketov. Primárnym zámerom pri návrhu ulogd však bolo zaznamenávanie určitej skupiny paketov (napr. pri zisťovaní porušovania pravidiel) a nie je optimalizovaný na rýchlosť.

2.2.2 Libpcap

Pri zachytávaní pomocou knižnice libpcap prechádza paket rovnakým procesom spracovania, aký bol popísaný na začiatku kapitoly 2.2 s jedným rozdielom – ovládač sieťovej karty pošle kópiu každého prijatého a odoslaného paketu časti jadra operačného systému, kde sa nachádza príslušný paketový filter. Ten v základnom nastavení púšťa všetky pakety ďalej, ale môže byť nastavený tak, že pakety filtruje na základe zadaných pravidiel.

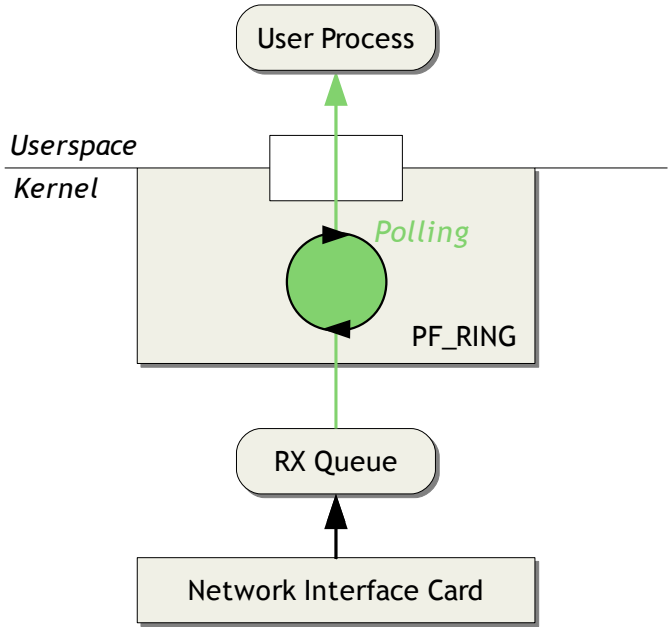
Knižnica libpcap poskytuje vysokoúrovňové rozhranie pre prístup k zachytávaným paketom, ktoré je nezávislé na použitej platforme. Toto rozhranie je navrhnuté pre použitie v jazykoch C a C++ a obsahuje funkcie na zistenie dostupných sieťových rozhraní, prístup k nim, získavanie paketov a funkcie pre nastavenie filtrovacích pravidiel pre použitý filter (Berkley Packet Filter). Pri návrhu libpcap bol kladený dôraz na jednoduchosť použitia a rýchlosť zachytávania paketov. Stále však dochádza k požiadavke na prerušenie pri každom prijatom pakete, čo znamená pri vysokom množstve paketov časté prepínanie kontextu. Tým je výkon do značnej miery obmedzený a dochádza k zahadzovaniu paketov.

2.2.3 PF_RING

PF_RING je implementácia soketu, ktorý využíva k získavaniu paketov techniku nazývanú polling. Jej princíp spočíva v tom, že po tom ako sieťová karta obdrží paket a vygeneruje prerušenie, operačný systém prevezme kontrolu nad obsluhou karty a počas tejto doby ignoruje ďalšie prerušenia od nej. Miesto toho naplánuje úlohu, ktorá periodicky obsluhuje potreby karty. Takýmto spôsobom ušetríme jadru operačného systému množstvo času stráveného obsluhou prerušení generovaných sieťovou kartou.

Ďalšou optimalizáciou, ktorú PF_RING zavádza pre zvýšenie rýchlosti je, že pakety nie sú ukladané do dátových štruktúr jadra operačného systému, odkiaľ sú následne zasielané užívateľskému procesu, ktorý si pakety vyžiadal. Miesto toho sú ukladané do kruhového buferu, ktorý si alokuje PF_RING pri vytvorení soketu. Každý paket vyzdvihnutý z vyrovnávacej pamäte sieťovej karty je uložený do kruhového buferu PF_RINGu odkiaľ ich priamo načítava užívateľský proces. Prichádzajúce pakety prepisujú v buferi tie, ktoré už boli vyzdvihnuté užívateľským procesom.

Nealokuje a nedealokuje sa žiadna nová pamäť. Detailný popis fungovania a optimalizácií, ktoré PF_RING používa sa nachádza v [13].



Obrázok 2.2: Architektúra PF_RING (inšpirované z [14]).

2.3 Vzorkovacie techniky

Pri meraní sieťovej premávky musíme počítať s prípadom, kedy výpočetné alebo pamäťové prostriedky, ktoré máme k dispozícií, nepostačujú k spracovávaniu objemu premávky tečúcej monitorovaným miestom. Takáto situácia môže nastať pri špičkách v prevádzke, kedy procesor nestíha analyzovať všetky pakety a priraďovať ich príslušným tokom. Ďalším prípadom sú rôzne sieťové útoky (napr. denial-of-service) alebo agresívne skenovanie portov, pri ktorých vzniká množstvo rozličných tokov a tie môžu ľahko zaplniť dátové štruktúry, do ktorých sa záznamy ukladajú. Jedným zo spôsobov, ako si s takouto situáciou poradiť, je spracovávať len vybrané pakety.

Zavedením vzorkovania paketov, prípadne znížením jeho frekvencie, znížime zaťaženie procesoru, množstvo ukladaných záznamov tokov a množstvo záznamov následne odosielaných na kolektor. Pre vzorkovanie paketov existujú rôzne spôsoby od jednoduchých až k sofistikovaným metódam, medzi ktoré patrí napríklad adaptívne vzorkovanie popísané v [6]. Táto metóda počíta optimálnu frekvenciu vzorkovania na základe aktuálneho objemu premávky a následne podľa nej renormalizuje uložené záznamy tokov. Komplexnosť týchto metód však prekračuje zameranie tejto práce. Zoznámime sa s niektorými základnými metódami vzorkovania, ktoré sú popísané v [4].

2.3.1 Systematické vzorkovanie

Systematické vzorkovanie spočíva v zvolení intervalu vzorkovania podľa deterministickej funkcie. To znamená, že vyberáme pakety s pravidelnými rozostupmi – buď v závislosti na priestorovej pozícií paketu alebo na základe časových intervalov. Príklad vzorkovania na základe pozície je výber každého K-tého prichádzajúceho paketu. Pri vzorkovaní na základe časových intervalov, môžme napríklad vybrať prichádzajúce pakety v intervale prvých 100 ms každú sekundu monitorovania (pakety, ktoré prišli počas zvyšných 900 ms sú zahodené).

Systematické vzorkovanie je jednoduché na implementáciu, ale nesie so sebou riziko skreslenia výsledkov. V prípade, že periodicita vzorkovania je podobná periodicite nejakého javu v sieťovej premávke, rastie pravdepodobnosť, že tento jav zostane vo výsledkoch merania skrytý.

2.3.2 Náhodné vzorkovanie n z N

Pri náhodnom vzorkovaní vyberáme pakety v súlade s náhodným procesom. K tomuto účelu potrebujeme pseudonáhodné čísla. Pri náhodnom vzorkovaní n z N je vybraných n paketov z celkového počtu N. Jedným zo spôsobov ako to dosiahnuť, je vygenerovanie n rôznych pseudonáhodných čísel z intervalu $<1,N>$ a následný výber paketov, ktoré sa nachádzajú na pozíciách prislúchajúcich týmto náhodným číslam. Počet vzoriek n je fixný (v jednotlivých periódach vzorkovania sa nemení).

2.3.3 Náhodné vzorkovanie s rovnomerným rozložením pravdepodobnosti

Jedná sa taktiež o metódu náhodného vzorkovania, avšak v tomto prípade je každý jednotlivý paket vybraný s pravdepodobnosťou p. Túto pravdepodobnosť určuje frekvencia vzorkovania. Implementácia spočíva vo vygenerovaní náhodného čísla n z intervalu $<0,1>$ s rovnomerným rozložením. Ak je $n \leq p$ paket je vybraný pre ďalšie spracovanie, v opačnom prípade je paket zahodený.

Náhodnými spôsobmi výberu paketov sa môžme vyhnúť problémom, ktoré môžu vzniknúť pri systematickom vzorkovaní a získať menej skreslené dáta. Ich použitie sa odporúča v monitorovacích zariadeniach, pretože majú dobré štatistické vlastnosti a sú nenáročné na implementáciu.

2.4 Indexácia

Po obdržaní paketu, ktorý má byť spracovaný, musíme nájsť záznam toku, ktorému tento paket patrí. Vzhľadom k tomu, že tento proces musíme vykonávať pre každý spracovávaný paket, kľúčovým je

rýchle vyhľadávanie v záznamoch. Dátovou štruktúrou, ktorá umožňuje obecne najrýchlejšie vyhľadávanie je hašovacia tabuľka.

Pre pripomenutie z kapitoly 2.1 – pakety náležia určitému toku na základe skupiny spoločných vlastností – kľúčových polí. Kľúčové polia označujeme ako identifikátor toku (jednoznačne ho definujú). Pre vyhľadanie záznamu určitého toku teda potrebujeme použiť všetky kľúčové polia. Vzhľadom k variabilite zvolených kľúčových polí majú tieto vo výsledku variabilnú dĺžku, ktorá môže naviac dosahovať stovky bitov. Preto nie je prakticky možné priamo mapovať identifikátor toku na jeho záznam a musíme použiť jeho haš.

2.4.1 Naivné hašovanie

Pri technike zvanej naivné hašovanie použijeme hašovaciu funkciu aby sme z identifikátoru toku získali haš pevnej dĺžky, ktorý potom použijeme ako index do tabuľky. Na danom mieste v tabuľke sa nachádza ukazovateľ na záznam s dátami toku. Ak je dané miesto v tabuľke prázdne, ide o nový tok, pre ktorý vytvoríme nový záznam. Ak v tabuľke už záznam existuje, aktualizujeme ho v závislosti na dátach obsiahnutých v pakete.

Vzhľadom k tomu, že dĺžka hašu je spravidla menšia ako dĺžka identifikátoru toku, z ktorého sa haš počíta, môže nastať situácia, kedy pre rôzne toky s rôznymi identifikátormi získame rovnaký haš. Tento jav sa nazýva kolízia. Pri naivnom hašovaní sú kolízie riešené tak, že rôzne záznamy s rovnakou hodnotou hašu sú uložené ako lineárny zoznam. Tým však stúpa časová zložitosť vyhľadávania z $O(1)$ na $O(n)$.

2.4.2 Cuckoo hašovanie

Cuckoo hašovanie umožňuje uložiť na každú pozíciu v tabuľke maximálne jeden záznam toku. Na výpočet hašu identifikátoru toku však máme k dispozícií dve rôzne hašovacie funkcie f_1 a f_2. Záznamy sú ukladané na miesto určené hašovacou funkciou f_1. V prípade vzniku kolízie pri vkladaní nového záznamu, je existujúci záznam zo svojho miesta vyhodený a uložený na alternatívnu pozíciu určenú hašovacou funkciou f_2. Ak alternatívna pozícia nie je voľná, postup opakujeme. Záznam, ktorý miesto používa vyhodíme a pre jeho uloženie použijeme jeho alternatívnu pozíciu. Tento postup sa opakuje, kým nie je nájdená voľná pozícia alebo kým proces hľadania neprekročil daný počet krokov. V druhom prípade sú zvolené nové hašovacie funkcie a celá tabuľka je pomocou nich rekonštruovaná (rehašovaná).

Pri Cuckoo hašovaní dosiahneme v najhoršom prípade konštantný čas vyhľadávania $O(2)$. Vkladanie je úspešné v konštantnom čase aj v prípade nutnosti rehašovať tabuľku, ak je vyťaženie tabuľky najviac 49% [16]. To do značnej miery limituje využitie kapacity tabuľky. Ak však Cuckoo

hašovanie zobecníme na použitie n alternatívnych hašovacích funkcií, efektívne využiteľná kapacita stúpa. Pri použití 3 hašovacích funkcií je možné tabuľku vyťaziť už na 91% [17].

V článkoch [18] a [19] je ukázané, že Cuckoo hašovanie je na dnešných procesoroch výrazne rýchlejšie ako konvenčné metódy hašovacích tabuliek, vďaka vysokému stupňu zásahov do pamäti cache.

2.5 Agregácia

Ďalším spôsobom, ako sa vyrovnať pri meraní tokov s nárokmi na prostriedky, ktoré prekračujú možnosti monitorovacieho zariadenia je agregácia. Techniky agregácie tokov znižujú objem monitorovaných dát zanedbaním niektorých informácií o toku a zoskupením tokov na základe určitých spoločných vlastností. Tým sa znížia pamäťové nároky merania a objem exportovaných dát. Výhodou agregácie je, že nestrácame žiadnu časť sieťovej premávky ako je tomu pri vzorkovaní paketov, ale získavame prehľad o celej premávke v nižšej úrovni detailov. Agregácia môže byť s výhodou použitá napríklad v nasledovných prípadoch [20]:

- pri archivácií nameraných dát môžme znížiť časové rozlíšenie spojením po sebe idúcich záznamov rovnakého toku
- v prípade klient-server aplikácií informáciu o danej službe väčšinou udáva číslo portu na strane servera, číslo portu na strane klienta preto môže byť zanedbané a nepoužíva sa ako kľúčové pole toku
- aplikácie vykonávajúce účtovanie na základe paketov potrebujú len informácie o objeme prijatých a odoslaných dát, prípadne dát vymenených dvoma sieťami na špecifickej bráne a informácie o jednotlivých tokoch v takomto prípade nie sú vôbec potrebné

Agregácia môže byť realizovaná sondou predtým, než sú dáta exportované alebo samostatným zariadením zvaným koncentrátor, ktorý zbiera dáta zo sond, tieto agreguje a následne odosiela kolektoru.

2.5.1 Adaptívna agregácia

V článku [21] je navrhnutá metóda adaptívnej agregácie tokov. Jej princíp spočíva v automatickom zlúčení tokov vo chvíli, keď sa začne zapĺňať pamäť, do ktorej sú ukladané záznamy tokov.

Agregácia môže prebiehať v 3 úrovniach: L1 – zlúčenie tokov, ktoré majú spoločné jedno kľúčové pole (napr. cieľovú IP adresu), L2 – zlúčenie tokov, ktoré majú spoločné 2 kľúčové polia (napr. zdrojovú a cieľovú IP adresu), L3 – zlúčenie tokov, ktoré majú spoločné 3 kľúčové polia (napr. zdrojovú IP adresu, cieľovú IP adresu, cieľový port). Pri použití úrovne L1 dochádza k najvýraznejšiemu zníženiu počtu tokov, ale úroveň detailov o tokoch je najnižšia.

Metóda používa komplexný algoritmus, ktorého cieľom je vyhľadať toky, ktoré budú agregované pričom:

1. počet uvoľnených záznamov po agregácií uspokojí pamäťové požiadavky a
2. úroveň agregácie bude čo najnižšia.

Ako dátovú štruktúru používa dvojrozmernú hašovaciu tabuľku. Jeden rozmer používa ako index haš vypočítaný zo zdrojovej IP adresy a indexom druhého rozmeru je haš vypočítaný z cieľovej IP adresy. Táto štruktúra ukladá toky, ktorých pravdepodobnosť neskoršieho zlúčenia je vyššia blízko seba, vďaka čomu je manipulácia so záznamami pri zlučovaní efektívna.

Výhody tejto metódy spočívajú v tom, že ani pri útokoch drasticky neznižuje vzorkovaciu frekvenciu a ani v prípade zaplnenia pamäte nezahadzuje legitímne toky, takže garantuje presnosť meraných dát. Vďaka tomu je táto metóda schopná poskytnúť správcom užitočné informácie pre analýzu prebiehajúcich útokov apod.

2.5.2 Flexibilná agregácia

Pri flexibilnej agregácií tokov navrhnutej v [20] sú dáta agregované na strane sondy na základe konfigurácie, ktorú sonde dodá operátor. Konfigurácia je prevádzaná tzv. agregačnými pravidlami.

Agregačné pravidlá špecifikujú ako má merací proces pri spracovaní paketu zaobchádzať s jednotlivými poliami a na ten účel používa 4 kľúčové slová – `discard`, `keep`, `mask/n`, `aggregate`. Prvé dve určujú, ktoré polia (kľúčové alebo nekľúčové) sú ponechané a ktoré sú zahodené a neobjavia sa vo výslednom zázname toku. Pre zdrojové a cieľové IP adresy môžu byť špecifikované masky, čím je dosiahnutá možnosť rozlišovať toky na základe celých sietí miesto jednotlivých IP adries. Kľúčové slovo `aggregate` slúži na označenie polí, ktorých štatistiky majú byť pri agregácií tokov spočítané (napr. počet paketov, počet bajtov, čas začiatku a konca toku).

Pre označenie polí v agregačných pravidlách sú použité názvy informačných elementov protokolu IPFIX [3]. Voliteľne môže byť pri každom poli v konfigurácií uvedený vzor, s ktorým je príslušná hodnota prichádzajúcich paketov porovnaná. Ak táto hodnota s uvedeným vzorom nesúhlasí paket nie je započítaný. Týmto spôsobom je možné filtrovať sieťovú premávku pri agregácií tokov. Príklad agregačného pravidla prevzatý z [20]:

```
Aggregation rule
    discard sourceTransportPort
    mask/24 sourceIpv4Address
    discard destinationTransportPort in 80, 443
    keep destinationIpv4Address in 10.10.10.0/16
    aggregate packetDeltaCount
```

Pri odosielaní agregovaných dát musíme pamätať na to, že kolektor, ktorý dáta obdrží a bude ich ďalej spracovávať nepozná pravidlá a podmienky, za ktorých boli dáta namerané. Vzhľadom k tomu, že protokoly pre export dát sú navrhnuté pre jednosmernú komunikáciu, kolektor nemá možnosť tieto informácie ani explicitne zistiť. Pri odosielaní preto musíme k nameraným dátam pripojiť aj informácie, potrebné k ich interpretácií.

2.6 Export dát

Po tom ako je dátový tok ukončený je potrebné jeho záznam odoslať príslušnému kolektoru, ktorý spracúva namerané dáta. Kolektor spravidla beží na vzdialenom zariadení a musí prijímať informácie o tokoch z viacerých miest v sieti. Dáta je preto potrebné doručovať po sieti a v jednotnej reprezentácií. Pre tento účel bolo navrhnutých niekoľko sieťových protokolov. Medzi najrozšírenejšie patrí protokol Netflow vyvíjaný firmou Cisco. Napriek tomu, že ide o proprietárny protokol, niektoré jeho verzie sa stali rozšírenými a sú implementované na rôznych platformách a zariadeniach iných výrobcov. Dnes najpoužívanejšie sú Netflow verzia 5, ktorá je však obmedzená len na export dát z IPv4 sietí a formát exportu je fixný a Netflow verzia 9, ktorá umožňuje definíciu šablón pre formát exportu a poradí si so záznamami rôznych sieťových protokolov ako IPv4, IPv6, MPLS [7]. Kvôli potrebe nezávislého a štandardizovaného exportovacieho protokolu vznikol protokol IPFIX (IP Flow Information Export), ktorý vychádza z Netflow verzia 9 a rozširuje ho o ďalšie vlastnosti.

2.6.1 IPFIX

Vývoj IPFIX zastrešuje pracovná skupina IETF (Internet Engineering Task Force). Momentálne existuje ako návrh na štandard a je popísaný v [1]. Vzhľadom k tomu, že export dát nezabezpečuje protokol ako taký, definícia IPFIX obsahuje okrem samotného formátu protokolu aj popis architektúry a požiadavky kladené na merací systém a pevne vymedzuje používané pojmy.

Architektúra

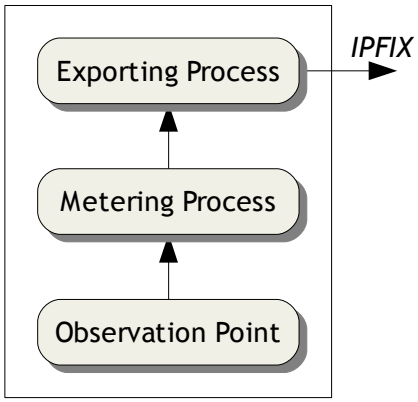
IPFIX definuje nasledovné 4 bloky, ktoré tvoria architektúru meracieho systému:

- sledovaný bod (observation point)
- merací proces (metering process)
- exportovací proces (exporting process)
- zberný proces (collection process)

Sledovaný bod je miesto v sieti, kde môžeme získavať IP pakety. Príkladom môžu byť linka, ku ktorej je pripojená sonda, sieťové rozhranie smerovača alebo zdielané médium ako napr.

ethernetová lokálna sieť. **Merací proces** vytvára samotné záznamy tokov. Do tohto procesu vstupujú pakety získaných v sledovanom bode. Merací proces obsahuje funkcie na získavanie hlavičiek paketov, vytváranie časových razítok paketov, vzorkovanie paketov, ich následnú klasifikáciu a funkcie na vyhľadávanie, aktualizáciu a správu tokov. **Exportovací proces** zabezpečuje odosielanie záznamov o tokoch, ktoré sú generované meracím procesom jednému alebo viacerým zberným procesom. **Zberný proces** prijíma záznamy tokov z jedného alebo viacerých exportovacích procesov. Ďalej tieto záznamy spracúva, prípadne ich môže ukladať na neskoršie spracovanie.

Obecná architektúra sondy používanej pre meranie tokov je zachytená na obrázku 2.3. Pozostáva z jedného sledovaného bodu, z jedného meracieho procesu a jedného exportovacieho procesu.



Obrázok 2.3: Bloková schéma sondy

V kapitole 2.1 bola uvedená obecne zaužívaná definícia toku. Naproti tomu v IPFIX je tok definovaný nasledovne:

Tok je množina IP paketov prechádzajúca sledovaným bodom siete v určitom časovom intervale. Pakety patriace do určitého toku majú množinu spoločných vlastností. Každá vlastnosť je definovaná ako výsledok aplikácie funkcie na hodnoty:

1. jedného alebo viacerých polí hlavičky paketu (napr. cieľová IP adresa), hlavičky paketu transportného protokolu (napr. cieľový port) alebo hlavičky paketu aplikačného protokolu
2. jednej alebo viacerých charakteristík samotného paketu (napr. MPLS značky)
3. jedného alebo viacerých polí získaných pri spracovaní paketu (napr. výstupné rozhranie)

Paket patrí do daného toku, ak spĺňa všetky definované vlastnosti toku.

Vlastnosti, na základe ktorých sa toky rozlišujú nazývame **kľúčové polia**. Skupina vlastností, ktorá má byť použitá na rozlišovanie príslušnosti paketu k určitému toku zavisí od nastavenia meracieho procesu. Zberný proces musí byť schopný identifikovať, ktorá skupina vlastností bola

použitá pre rozlíšenie každého prijatého záznamu toku od ostatných tokov. Merací proces musí byť schopný vyhodnotiť príslušnosť paketov k daným tokom podľa nasledovných vlastností:

- vstupné a výstupné rozhranie toku
- polia hlavičky IP paketu – 1. zdrojová IP adresa, 2. cieľová IP adresa, 3. typ protokolu (pre IP adresy musí byť podporované rozlišovanie aj podľa prefixu zadanej dĺžky)
- polia hlavičky protokolu transportnej vrstvy – 1. zdrojový port, 2. cieľový port TCP/UDP protokolu
- MPLS značka (ak sa sledovaný bod nachádza na zariadení podporujúcom protokol MPLS)
- DSCP (ak sa sledovaný bod nachádza na zariadení podporujúcom prioritizáciu podľa DiffServ Code Point v hlavičke IP paketu)

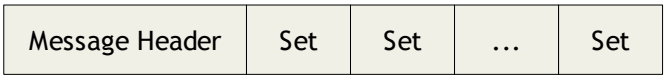
Požiadavky na merací proces ďalej zahŕňajú spoľahlivosť (žiadny prijatý paket nie je zahodený bez spracovania), vzorkovanie paketov, schopnosť vyrovnať sa s preťažením, generovanie časového razítka prvého a posledného prijatého paketu každého toku, synchronizáciu časových razítok s UTC (Coordinated Universal Time) a expiráciu toku pri neaktivite (neaktívny timeout).

Pre export dát IPFIX definuje informačný a dátový model. Informačný model (popísaný v [3]) obsahuje zoznam atribútov, ktoré môžme pre daný tok zaznamenávať a popisuje tiež ich sémantiku. Tieto atribúty sa nazývajú **informačné elementy**. Informačné elementy štandardizované IETF majú identifikátory pridelené od IANA (Internet Assigned Numbers Authority). Autorizované spoločnosti môžu definovať vlastné informačné elementy. Tieto sú od štandardizovaných odlíšené tým, že najvýznamnejší bit v poli informačného elementu má hodnotu 1. Dátový model definuje, ako sú zaznamenávané dáta reprezentované.

Kvôli flexibilite záznamu je špecifikáciou vyžadované, aby bol merací a exportovací proces konfigurovateľný užívateľom. Merací proces by mal umožňovať zadanie sledovaného bodu (napr. sieťové rozhranie), špecifikáciu tokov, ktoré majú byť merané, nastavenie aktívneho a neaktívneho timeoutu. Exportovací proces by mal umožňovať nastavenie formátu odosielaných záznamov (vrátane výberu atribútov toku, ktoré má záznam obsahovať) a špecifikáciu zberných procesov, ktorým majú byť zasielané dáta Ak je konfigurácia robená vzdialene, tento prenos by mal byť zabezpečený.

Formát správy

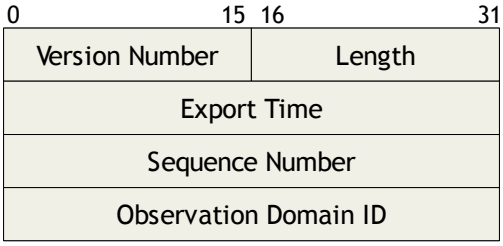
Správa IPFIX protokolu pozostáva z hlavičky nasledovanej jedným alebo viacerými setmi. Existujú 3 rôzne typy setov – dátový set, set šablóny a set šablóny volieb, ktoré budú popísané ďalej v tejto kapitole. Formát správy je zobrazený na obrázku 2.4.



Obrázok 2.4: Formát IPFIX správy

Maximálna dĺžka IPFIX správy je 65 535 bajtov, vzhľadom na obmedzenie poľa udávajúceho jej dĺžku na 16 bitov. **Hlavička správy** zobrazená na obrázku 2.5 má pevnú dĺžku 16 bajtov a obsahuje nasledovných 5 polí:

- číslo verzie (version number) – verzia formátu správy, pre terajšiu verziu IPFIX je to vždy hodnota 0x000a
- dĺžka (length) – celková dĺžka správy včetne hlavičky, udávaná v bajtoch
- čas exportu (export time) – čas, kedy paket opustil exportovací proces (v sekundách, počítaný od 1. januára 1970 00:00)
- sekvenčné číslo (sequence number) – sekvenčné počítadlo modulo 2^{32}, ktoré sa zvyšuje vždy o počet dátových záznamov odoslaných v danej správe, šablóny a šablóny volieb toto počítadlo neinkrementujú
- ID sledovanej oblasti (observation domain ID) – 32-bitový identifikátor, unikátny v rámci každého exportovacieho procesu. Je používaný zberným procesom na rozlíšenie, „kto“ zaznamenal daný tok v prípade, že exportovací proces odosiela dáta získané z viacerých zdrojov. Odporúča sa, aby bol unikátny aj v rámci každého IPFIX zariadenia.



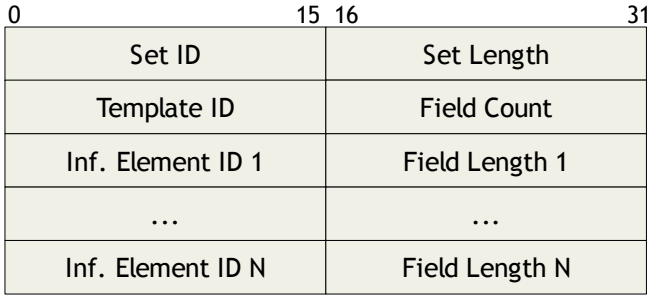
Obrázok 2.5: Formát hlavičky IPFIX správy

Pojem set sa používa pre označenie skupiny záznamov, ktoré majú rovnakú štruktúru. Každý set pozostáva z hlavičky setu a jedného alebo viacerých záznamov. Hlavička je pre všetky 3 typy setov spoločná a obsahuje 2 polia:

- ID setu (set ID) – 16-bitové číslo, ktoré identifikuje typ setu. Hodnota 2 je rezervovaná pre set šablóny, hodnota 3 pre set šablóny volieb a hodnoty 4 – 255 sú rezervované pre budúce použitie. Hodnoty nad 255 je možné používať pre označovanie dátových setov
- dĺžka setu (set length) – celková dĺžka setu udávaná v bajtoch, vrátane hlavičky setu, všetkých záznamov a prípadného zarovnania

Set šablóny obsahuje ľubovolnú kombináciu identifikátorov informačných elementov, čím definuje šablónu pre dáta zasielané v dátových setoch. Nepotrebujeme vďaka tomu vkladať k dátovým záznamom žiadne informácie k ich interpretácií. Zberný proces dokáže dáta interpretovať

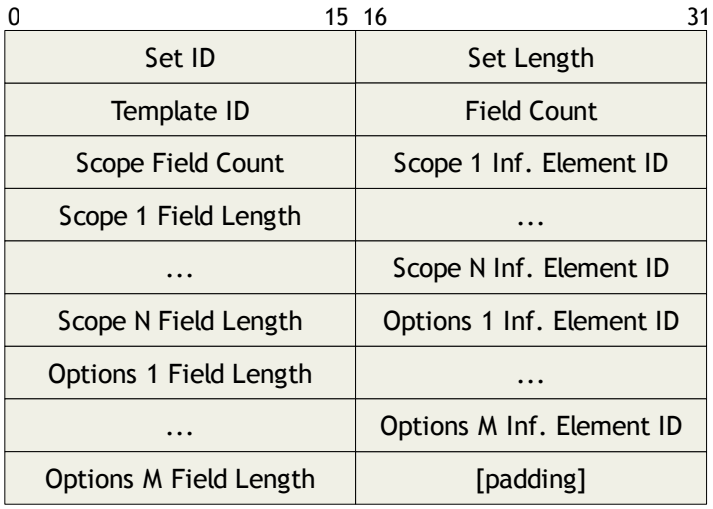
na základe šablóny, ktorú obdržal predtým prípadne ešte len obdrží v budúcnosti. Formát setu šablóny je na obrázku 2.6.



Obrázok 2.6: Formát setu šablóny

- ID šablóny (template ID) – identifikátor pridelený každej novej šablóne, je unikátny v rámci sledovanej oblasti (observation domain), musí byť väčší ako 255
- počet polí (field count) – počet informačných elementov obsiahnutých v tejto šablóne
- ID informačného elementu (inf. element ID) – numerická hodnota identifikujúca druh informačného elementu
- dĺžka poľa (field length) – dĺžka príslušného informačného elementu v bajtoch

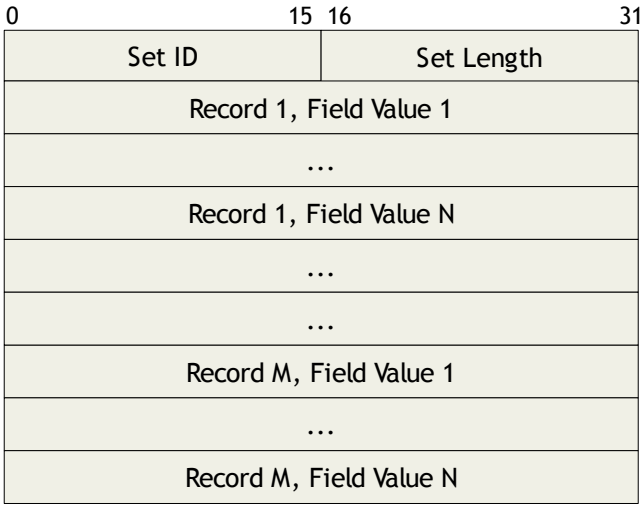
Set šablóny volieb umožňuje zaslať zbernému procesu dodatočné informácie, ktoré nie je možné doručiť pri zasielaní záznamov tokov prostredníctvom dátového setu a setu šablóny. Medzi takéto informácie patria napríklad kľúčové polia toku, štatistiky spoľahlivosti meracieho procesu (počet nespracovaných paketov) a podobne. Na tento účel používa elementy udávajúce rozsah platnosti bežných elementov. Formát setu šablóny volieb je na obrázku 2.7.



Obrázok 2.7: Formát setu šablóny volieb

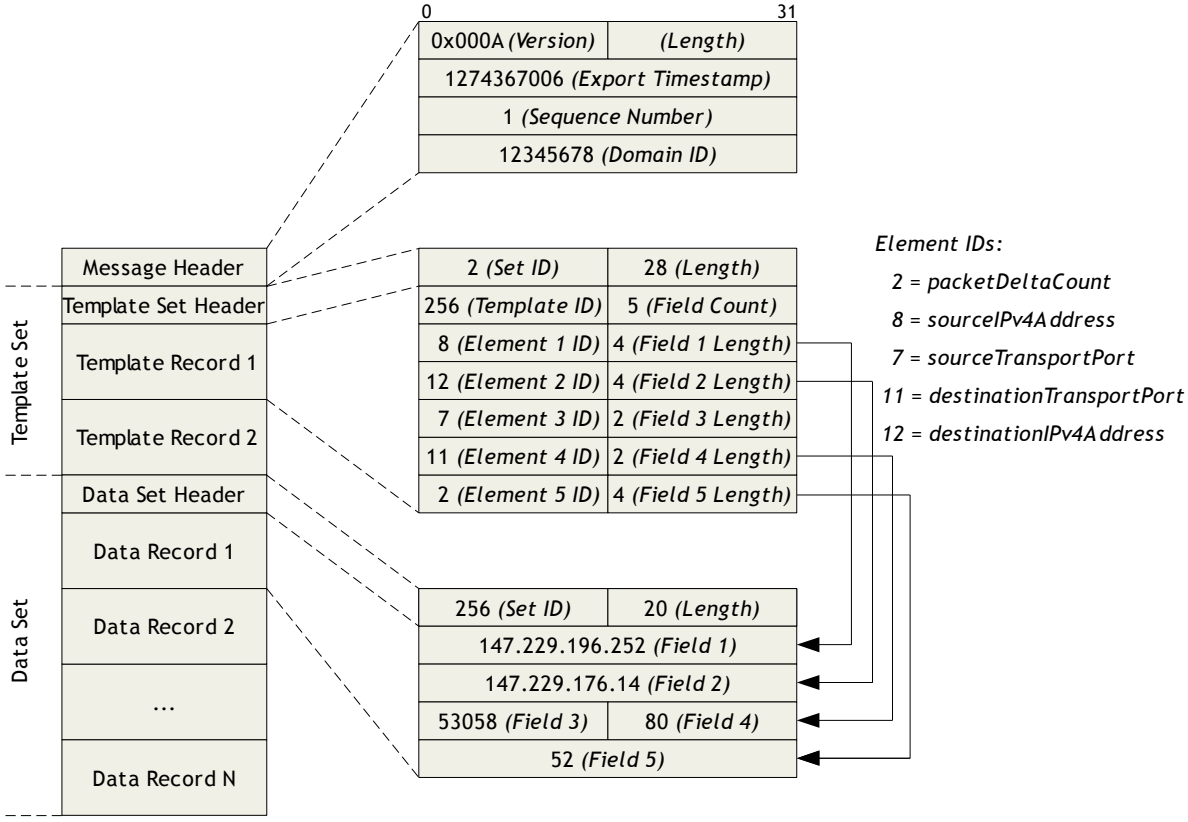
- ID šablóny (template ID) – identifikátor pridelený každej novej šablóne, je unikátny vzhľadom na sledovanú oblasť (observation domain), musí byť väčší ako 255
- počet polí (field count) – celkový počet informačných elementov obsiahnutých v tejto šablóne, v našom príklade N+M položiek
- počet polí udávajúcich rozsah platnosti šablóny (scope field count) – počet polí (nasledujúcich bezprostredne za týmto), ktoré obsahujú informačné elementy udávajúce rozsah platnosti šablóny, v našom prípade N položiek
- ID informačného elementu udávajúceho rozsah platnosti (scope inf. element ID) – numerická hodnota identifikujúca niektorý z informačných elementov, ktorý špecifikuje rozsah platnosti šablóny (napr. ID karty, na ktorej boli dáta namerané, ID meracieho procesu, ID šablóny, ...)
- dĺžka poľa (scope field length) – dĺžka príslušného informačného elementu udávajúceho rozsah platnosti
- ID informačného elementu (information element ID) – numerická hodnota identifikujúca typ bežného informačného elementu
- dĺžka poľa (field length) – dĺžka príslušného informačného elementu v bajtoch

Dátový set slúži na zasielanie nameraných dát. Obsahuje jeden alebo viac záznamov rovnakého typu, ktoré reprezentujú dáta nazbierané o jednotlivých tokoch. Štruktúra týchto záznamov je definovaná setom šablóny alebo setom šablóny volieb. Identifikátor šablóny, podľa ktorej majú byť záznamy reprezentované je uvedené v dátovom sete v poli ID setu. Identifikátory šablón dátových setov môžu nadobúdať hodnoty 256 – 65 535. Formát dátového setu je na obrázku 2.8.



Obrázok 2.8: Formát dátového setu

Jednotlivé typy setov sa môžu v zaslanej správe vyskytovať v ľubovolnom poradí a ich množstvo je obmedzené iba maximálnou dĺžkou IPFIX správy. Každý set môže tiež obsahovať ľubovolné množstvo záznamov rovnakého typu. Sety šablón nesmú byť zasielané opakovane s každým príslušným dátovým setom. Sú zasielané zbernému procesu iba raz – pred alebo súčasne so zaslaním prvého dátového setu, ktorý má byť podľa danej šablóny interpretovaný. Príklad IPFIX správy zobrazený pre názornosť v dvoch úrovniach je na obrázku 2.9.



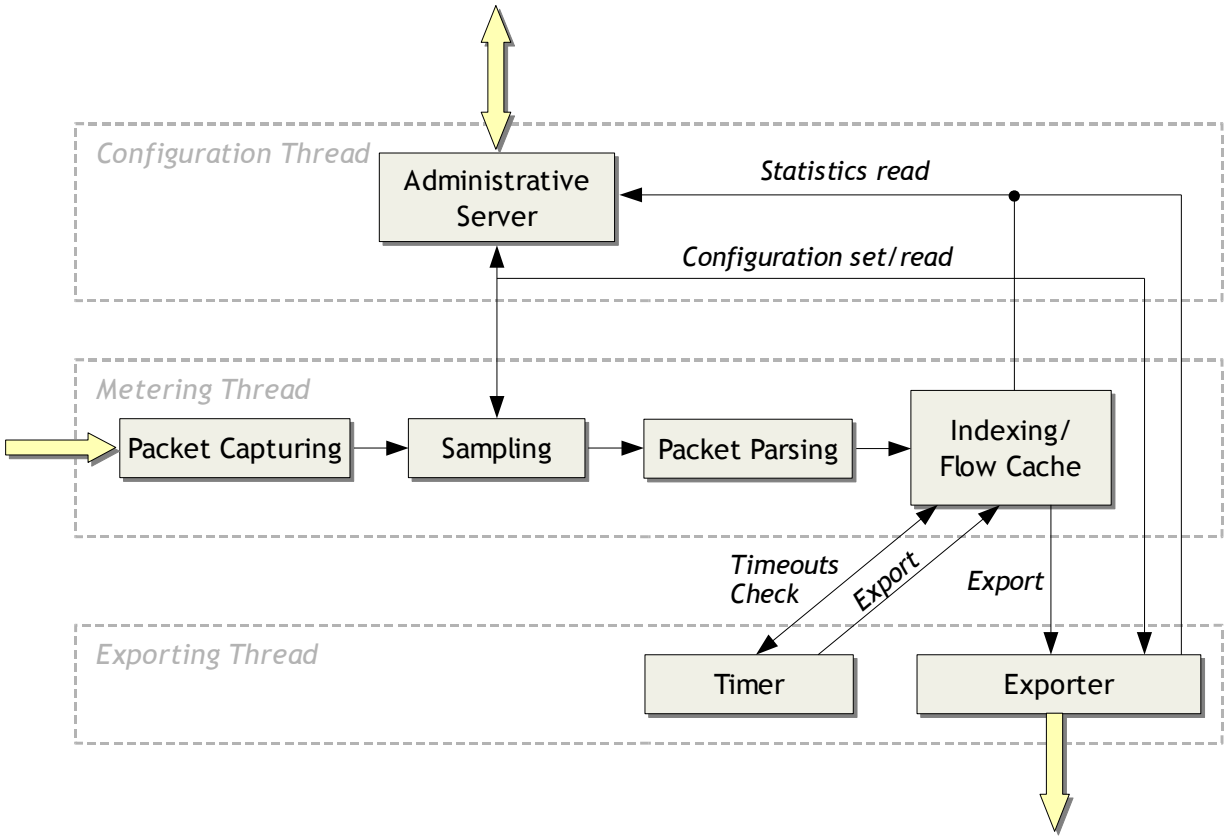
Obrázok 2.9: Viacúrovňová schéma IPFIX správy

IPFIX správa je zabalená do protokolu transportnej vrstvy, pričom IPFIX je navrhnutý tak, aby bol nezávislý na použitom transportnom protokole. Špecifikácia vyžaduje, aby bol implementovaný prenos protokolom SCTP, ktorý umožňuje spoľahlivý prenos a ochranu proti zahlteniu prenosového pásma. Voliteľne môže byť naviac implementovaný prenos protokolom UDP, ktorý zabezpečuje nespoľahlivý prenos a prenos protokolom TCP, ktorý zabezpečuje spoľahlivý prenos dát.

3 Návrh architektúry sondy

Predmetom tejto kapitoly je návrh softvérovej sondy pre meranie sieťových tokov. Pri návrhu je kladený dôraz na flexibilitu záznamu toku, efektívnu indexáciu záznamov a rýchlosť spracovania dát, od ktorej sa odráža celkový výkon sondy a jej schopnosť spoľahlivo merať premávku na dnešných vysokorýchlostných sieťach. Ďalšou požiadavkou je možnosť zistenia aktuálneho stavu merania a zmena nastavení sondy počas jej behu. Pre export nameraných dát bol zvolený protokol IPFIX, popísaný v podkapitole 2.6.

V kapitole 2 je popísaný postup a princípy používané pri meraní tokov. Tento postup môže byť rozdelený do funkčných blokov vykonávajúcich jednotlivé dielčie úlohy merania. Tie spolu s ďalšími požiadavkami kladenými na sondu, ktorej návrh je predmetom tejto práce, tvoria základ jej architektúry zobrazenej na obrázku 3.1.



Obrázok 3.1: Bloková schéma sondy

Funkčné bloky sondy:

- jednotka zachytávania paketov (Packet Capturing) – zabezpečuje získavanie paketov v sledovanom bode. Tie môžu byť následne (v závislosti od nastavenia) vzorkované.
- jednotka pre extrakciu dát z paketu (Packet Parsing) – extrahuje potrebné polia z hlavičiek paketov a získanými dátami naplní štruktúry potrebné pre aktualizáciu záznamu príslušného

toku. Jej súčasťou je aj výpočet hodnoty hašu kľúčových polí paketu. Kľúčové polia paketu sú tie isté kľúčové polia, ktoré identifikujú príslušný sieťový tok (viď. definícia pojmov v podkapitole 2.6.1). Tento haš slúži pre následnú indexáciu záznamov tokov.

- indexačná jednotka (Indexing/Flow Cache) – zabezpečuje vyhľadávanie a aktualizáciu záznamov tokov. Obsahuje tiež pamäť, v ktorej sú záznamy a všetky s nimi súvisiace dáta uchovávané až do doby, kým nedôjde k ich expirácií a následnému odoslaniu na kolektor.
- exportér (Exporter) – zabezpečuje odoslanie záznamov expirovaných tokov kolektoru. Jeho úlohou je generovať zo záznamov tokov obdržaných od meracieho procesu dáta vo formáte IPFIX, ktoré sú potom zariadenia podporujúce tento formát schopné interpretovať. Chod exportéru je závislý na časovači (Timer). Ten v pravidelných intervaloch kontroluje aktuálnosť záznamov tokov a ak zistí, že niektorý z nich expiroval, dáva meraciemu procesu pokyn k jeho odoslaniu do exportéru.
- administračný server (Administrative Server) – jednou z požiadaviek na sondu je získavanie dát o priebehu merania a zmena niektorých jej parametrov počas merania. Z toho dôvodu potrebujeme mať možnosť nadviazať so sondou komunikáciu počas jej behu. Pre tento účel je súčasťou sondy administračný server, ktorý čaká na spojenie na zadanom porte a obsluhuje požiadavky zadávané operátorom prostredníctvom obslužného programu.

Beh programu je rozdelený do troch vlákien. V hlavnom vlákne (Metering Thread) sa prevádzajú úlohy odpovedajúce meraciemu procesu architektúry IPFIX, tzn. zachytávanie paketov a ich vzorkovanie, extrakcia dát paketov a indexovanie vznikajúcich záznamov tokov. Druhé vlákno odpovedá exportovaciemu procesu a v treťom vlákne beží administračný server sondy. Dôvodom pre použitie vlákien v návrhu sondy je asynchrónny beh administračného serveru a exportéru vzhľadom na merací proces. Administračný server čaká na spojenie prichádzajúce od obslužného programu sondy a to nastáva nezávisle na stave, v ktorom sa práve nachádza merací proces. Následná obsluha požiadavkov operátora nám vďaka behu serveru v oddelenom vlákne nebude spôsobovať časové oneskorenia prípadne výpadky v meracom procese. Činnosť exportéru je riadená časovačom, ktorého beh je tiež asynchrónny vzhľadom k meraciemu procesu. Vďaka časovaču sú záznamy tokov odosielané v pravidelných časových intervaloch po ich expirácií. Odosielanie veľkého množstva záznamov zo sondy, ktoré môže byť vzhľadom k prenosu po sieti časovo náročné, nám vďaka behu exportéru v nezávislom vlákne taktiež nebude blokovať činnosť meracieho procesu. Pri behu na viacprocesorovom systéme (prípadne na systéme s viacjadrovým procesorom) môže naviac každé vlákno aplikácie bežať na samostatnom procesore resp. jadre.

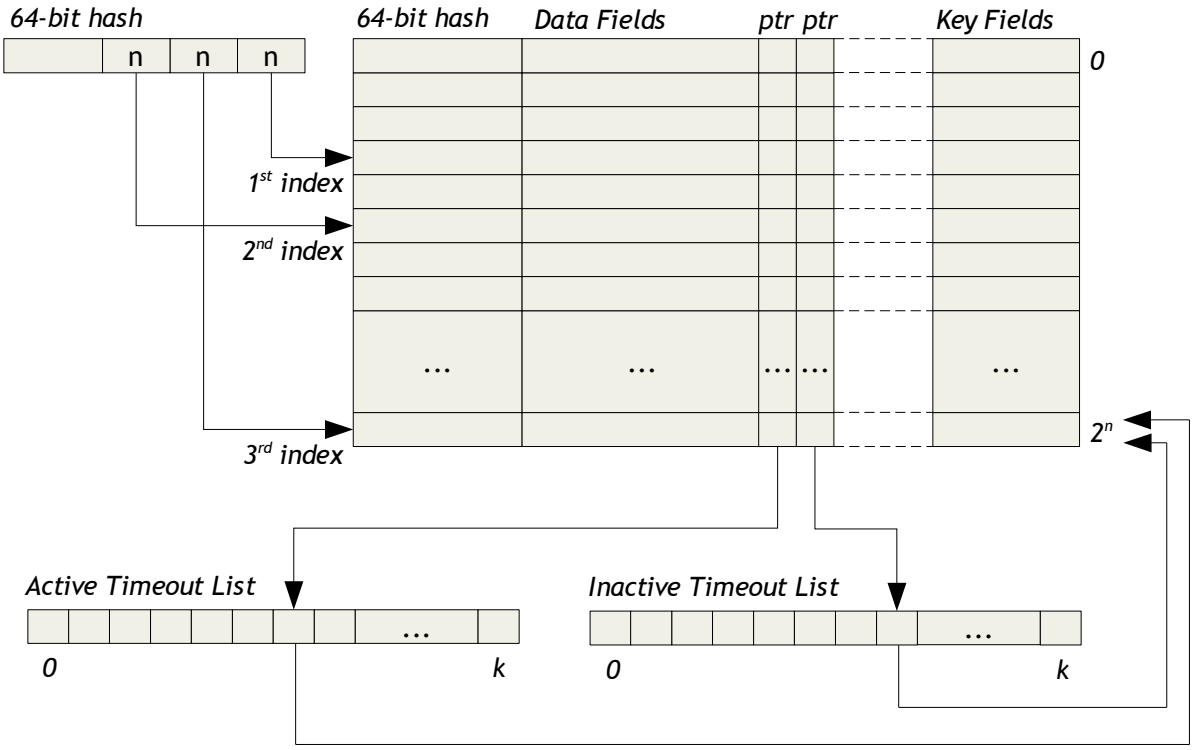
3.1 Algoritmy a dátové štruktúry

K zachytávaniu paketov je použitý soket PF_RING, popísaný v podkapitole 2.2, pre jeho najvyššiu dosahovanú rýchlosť z pomedzi dostupných riešení [13]. PF_RING tiež implementuje systematické vzorkovanie, ktoré je vykonávané priamo v jadre operačného systému, vďaka čomu je výrazne efektívnejšie ako vzorkovanie paketov v aplikácií (viď. podkapitola 5.2.3). Všetky pakety obdržané po vzorkovaní sú posunuté do ďalšieho spracovania jednotkou pre extrakciu dát. Tá volaním užívateľom definovanej funkcie (viď. podkapitola 3.2.2) naplní štruktúru obsahujúcu kľúčové polia

toku a štruktúru obsahujúcu dátové položky toku dátami extrahovanými z hlavičky paketu. Z kľúčových polí toku je vypočítaný haš dĺžky 64-bitov, ktorý je neskôr použitý indexačnou jednotkou pre vyhľadanie záznamu toku, ktorému aktuálny paket náleží. Dátové položky sú potom použité pre aktualizáciu dátovej časti záznamu toku.

3.1.1 Pamäť záznamov tokov

Pre ukladanie záznamov tokov v indexačnej jednotke je použitá dátová štruktúra, ktorá je zobrazená na obrázku 3.2. Pozostáva z dvoch oddelených tabuliek, ktoré obsahujú 2^n položiek, kde n je užívateľom nastaviteľný parameter, od ktorého je odvodený maximálny možný počet súčasne udržiavaných záznamov v pamäti sondy (tj. počet položiek v tabuľkách). Hodnota n pritom udáva počet bitov z hašu kľúčových polí, ktoré sú použité ako index do tabuľky. Spôsob vyhľadávania v tabuľke je popísaný v nasledujúcej podkapitole.



Obrázok 3.2: Dátová štruktúra pre ukladanie záznamov o tokoch

Dve tabuľky sú použité z dôvodu oddeleného ukladania kľúčových polí (tie sú do tabuľky vložené pri vytvorení záznamu toku a pri následnom aktualizovaní záznamu sa nemenia) od dátových polí, ktoré sú aktualizované s každým novým paketom náležiacim toku. To by malo zabezpečiť vyššiu úspešnosť zásahov vo vyrovnávacej pamäti procesoru. Tabuľka kľúčových polí (Key Fields)

obsahuje výhradne kľúčové polia toku. Tabuľka ukladajúca dátové položky toku (Data Fields) naviac obsahuje hodnotu hašu kľúčových polí o dĺžke 64-bitov a dva ukazovatele – jeden do zoznamu spravujúceho aktívne timeouty a druhý do zoznamu spravujúceho neaktívne timeouty. V týchto zoznamoch sú udržiavané čas vzniku toku a čas poslednej aktualizácie toku. Sú používané na kontrolu expirácie záznamu a ich použitie bude bližšie vysvetlené v podkapitole 3.1.4.

3.1.2 Vyhľadanie záznamu toku

Vkladanie a riešenie kolízií záznamov v tabuľke je realizované použitím Cuckoo hašovania, ktorého princíp je popísaný v podkapitole 2.4.2. Jeho základná verzia používa pre ukladanie záznamu do tabuľky 2 alternatívne pozície získané dvoma rôznymi hašovacími funkciami. Aby sme zvýšili efektívne využiteľnú kapacitu tabuľky budeme pre ukladanie záznamov tokov používať 3 alternatívne pozície, ktoré naviac získame výpočtom jedného 64-bitového kľúča. Z tohto kľúča následne použijeme prvých n bitov ako prvý index do tabuľky, druhých n bitov ako druhý index do tabuľky a tretích n bitov ako tretí index (viď. obrázok 3.2). Hodnota n je variabilná a stanovuje počet položiek tabuľky na 2^n. Spôsob získavania indexov, tzn. rozdelenie 64-bitového kľúča na 3 časti o dĺžke n bitov, obmedzuje maximálnu hodnotu n na 21 a tým maximálny počet položiek tabuľky na 2 097 152.

Toto riešenie nám zaručuje konštantnú časovú zložitosť O(3) pri vyhľadávaní toku. To, či sa na danej pozícií nachádza hľadaný záznam kontrolujeme porovnaním aktuálneho 64-bitového kľúča s kľúčom uloženým v tabuľke. Po tom, ako je odpovedajúci záznam nájdený, sú aktualizované dátové polia záznamu toku a údaj o jeho poslednej aktualizácií v zozname spravujúcom neaktívne timeouty.

Pre výpočet hodnoty 64-bitového hašu kľúčových polí toku bola na základe experimentov uvedených v kapitole 5.2.1 a analýz vlastností uvedených v článkoch [22] a [23] vybraná hašovacia funkcia navrhnutá Bobom Jenkinsom. Jej vstupom je pole bajtov ľubovolnej dĺžky a výstupom jeho haš o fixnej dĺžke 32-bitov, prípadne 2 rôzne hodnoty o dĺžke 32-bitov. Používame verziu, ktorá vracia 2 32-bitové hodnoty a ich konkatenáciou získame potrebný 64-bitový kľúč.

3.1.3 Vkladanie záznamu toku

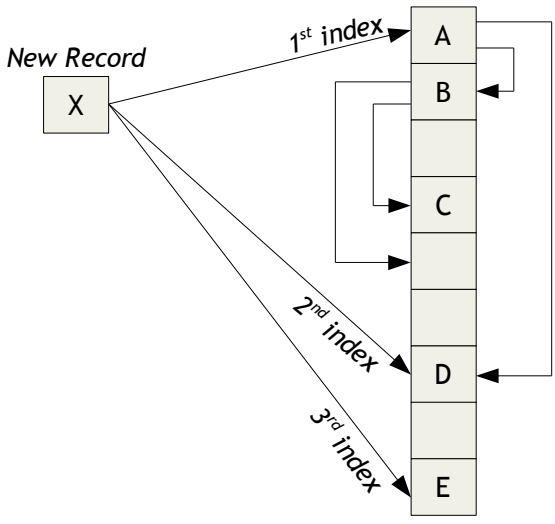
V prípade, že sa záznam toku v tabuľke nenachádza, tzn. nenašli sme ho na žiadnej z jeho troch alternatívnych pozícií, pokúsime sa ho do tabuľky vložiť. Označme p0(x), p1(x) a p2(x) pozície, na ktoré je možné uložiť záznam toku x. Vkladanie záznamu x potom prebieha v nasledovných krokoch:

1. skontroluj obsadenosť p0(x), p1(x) a p2(x) a vlož x na prvú voľnú pozíciu podľa vymenovaného poradia

2. ak sú všetky 3 pozície obsadené, vyhoď záznam y okupujúci p0(x) a na uvoľnené miesto vlož x
3. nech p[k](y) je pozícia, z ktorej sme vyhodili záznam y. Skontroluj obsadenosť p[(k + 1) modulo 3](y) a p[(k + 2) modulo 3](y).
4. ak je niektorá z nich voľná, použi ju na uloženie y. V opačnom prípade vyhoď záznam w okupujúci p[(k + 1) modulo 3](y)
5. pokračuj krokom č. 3 so záznamom w

Pri tomto postupe samozrejme môžu nastať situácie, keď pokus o vloženie cyklí do nekonečna. Z toho dôvodu zavádzame limit na počet presunutých záznamov. Po jeho prekročení je posledný záznam toku, ktorý sa nám nepodarilo do tabuľky umiestniť, násilne expirovaný a odoslaný na kolektor.

Postup vkladania je demonštrovaný na abstrahovanom príklade nachádzajúcom sa na obrázku 3.3. Najskôr sú skontrolované všetky alternatívne pozície nového záznamu X. Všetky sú obsadené (záznamami A, D a E), preto je z prvej pozície vyhodený záznam A a na uvoľnené miesto je vložený záznam X. V ďalšom kroku skontrolujeme obsadenosť dvoch zostávajúcich pozícií záznamu A. Obe sú obsadené (záznamami B a D) a preto z prvej z dvoch možných pozícií vyhodíme záznam B a na uvoľnené miesto vložíme A. V ďalšom kroku skontrolujeme obsadenosť dvoch zostávajúcich pozícií záznamu B. Druhý z nich je voľný, na toto miesto vkladáme B a končíme proces vkladania záznamu X.



Obrázok 3.3: Vkladanie záznamu použitím Cuckoo hašovania pre riešenie kolízií

3.1.4 Údržba záznamov

U záznamov existujúcich v pamäti sondy musíme vykonávať kontrolu validity. Na základe výsledkov tejto kontroly buď považujeme tok za stále platný a pokračujeme v držaní jeho záznamu v pamäti alebo tok považujeme za ukončený a jeho záznam expirujeme. Expirované záznamy sú následne predávané na spracovanie exportéru. Tok považujeme za ukončený ak nastane aspoň jedna z 3 nasledovných možností:

- jeho záznam existuje dlhšie ako stanovený časový interval (hodnota aktívneho timeoutu)
- jeho záznam nebol aktualizovaný novými dátami dlhšie ako je stanovený časový interval (hodnota neaktívneho timeoutu)
- obdržali sme paket s príznakom signalizujúcim ukončenie toku (TCP príznaky FIN a RST)

K detekcií týchto 3 prípadov potrebujeme mať k dispozícií nasledovné informácie: čas vloženia záznamu toku, čas poslednej aktualizácie toku a TCP príznaky aktuálneho paketu. Posledný z údajov nemusíme uchovávať, pretože hneď po jeho zistení v aktuálnom pakete môžme príslušný záznam expirovať. Pre uchovávanie zostávajúcich dvoch údajov bola dátová štruktúra reprezentujúca pamäť sondy rozšírená o obojsmerne viazané lineárne zoznamy (viď. obrázok 3.2). Do jedného z nich je pri vkladaní nového záznamu do tabuľky vložený prvok s časom vzniku záznamu a do druhého je vložený prvok s časom poslednej aktualizácie záznamu (ten je potom aktualizovaný s každou aktualizáciou záznamu toku). Každý prvok zoznamu obsahuje okrem časového údaju naviac index príslušného záznamu v tabuľke záznamov tokov.

Kontrola validity je možná sekvenčným prechádzaním a kontrolou všetkých prvkov v oboch lineárnych zoznamoch. Tento prístup má však lineárnu časovú zložitosť, takže sa pri vyššom počte záznamov stáva veľmi neefektívnym. Výhodnejšie je preto prvky v zoznamoch radiť chronologicky podľa časových údajov, ktoré obsahujú. U zoznamu, ktorý obsahuje čas poslednej aktualizácie tokov to znamená presunutie prvku práve aktualizovaného toku na koniec zoznamu. Vďaka tomu sa budú na začiatku zoznamu objavovať vždy najdlhšie neaktualizované záznamy. Zoznam obsahujúci čas vzniku toku je implicitne vždy chronologicky zoradený. Prvok obsahujúci čas vzniku nového toku je pridaný na koniec zoznamu a na jeho začiatku sa tak vždy objavujú prvky reprezentujúce najstaršie aktívne toky.

Keď máme zoznamy s časmi vzniku a poslednej aktualizácie toku chronologicky zoradené, stačí periodicky porovnávať časové záznamy na vrchole oboch zoznamov voči hodnote aktívneho resp. hodnote neaktívneho timeoutu. Túto činnosť zabezpečuje časovač, ktorý beží v exportovacom vlákne (viď. obrázok 3.1), aby svojou činnosťou neblokoval beh meracieho procesu. V prípade, že sú hodnoty niektorého z timeoutov prekročené, časovač dáva indexačnej jednotke pokyn k expirácií záznamu. Ten je následne indexačnou jednotkou predaný exportéru.

3.2 Flexibilita záznamu toku

Jednou z hlavných motivácií pre vznik navrhovanej sondy je flexibilita záznamu toku. To znamená, že je plne v réžii užívateľa:

- aké kľúčové polia budú použité pre rozlišovanie jednotlivých tokov
- aké ďalšie atribúty budú pre každý tok merané a uchovávané
- akými hodnotami budú kľúčové polia a atribúty naplnené pri príchode nového paketu
- akým spôsobom budú aktualizované atribúty toku, ktorému príchodzí paket náleží

Nastavenia týchto parametrov zároveň explicitne určujú formát záznamu toku, ktorý je následne odoslaný zo sondy na kolektor.

3.2.1 Konfigurácia formátu záznamu

Konfigurácia formátu záznamu je vykonávaná pred samotným prekladom aplikácie formou definície položiek štruktúr v hlavičkovom súbore zdrojových kódov. Užívateľ definuje položky dvoch dátových typov štruktúra:

- v štruktúre `key_fields` definuje kľúčové polia toku
- v štruktúre `data_fields` definuje ostatné (nekľúčové) atribúty toku

Definícia položky pozostáva zo špecifikácie typu informačného elementu IPFIX a identifikátoru tohto elementu, ako sú definované v informačnom modeli protokolu IPFIX [3]. Jednotlivé položky sú od seba oddelené bodkočiarkou (implementačne špecifická záležitosť závislá na zvolenom jazyku). Nasleduje príklad definície kľúčových položiek toku:

```
struct key_fields {
    unsigned8   protocolIdentifier;
    ipv4Address sourceIPv4Address;
    ipv4Address destinationIPv4Address;
    unsigned16  sourceTransportPort;
    unsigned16  destinationTransportPort;
}
```

Použitím tejto konfigurácie budú toky rozlišované na základe protokolu uvedeného v IP hlavičke, zdrojovej a cieľovej IP adresy a zdrojového a cieľového portu transportného protokolu. Príklad definície ďalších (nekľúčových) atribútov toku:

```
struct data_fields {
    dateTimeSeconds flowStartSeconds;
    dateTimeSeconds flowEndSeconds;
    unsigned64      octetDeltaCount;
}
```

Nekľúčovými atribútmi toku budú v prípade tejto konfigurácie časové razítko začiatku a konca príslušného toku (v sekundách) a súčet veľkostí všetkých paketov náležiacich toku od posledného hlásenia o danom toku (v bajtoch).

Okrem špecifikácie položiek v štruktúrach je nutné, aby užívateľ inicializoval pole kľúčových položiek a pole atribútov toku príslušnými identifikátormi použitých IPFIX elementov s prefixom `e_` pre interné použitie sondou. Príklad odpovedajúci vyššie definovaným štruktúram:

```
u_int16_t key_fields_array[] = {e_protocolIdentifier,
                                e_sourceIPv4Address,
                                e_destinationIPv4Address,
                                e_sourceTransportPort,
                                e_destinationTransportPort};

u_int16_t data_fields_array[] = {e_flowStartSeconds,
                                 e_flowEndSeconds,
                                 e_octetDeltaCount};
```

3.2.2 Extrakcia dát z paketu

Keďže atribúty toku sú definované užívateľom, je tiež nevyhnutné, aby mal užívateľ možnosť špecifikovať, akými dátami budú tieto atribúty naplnené. K tomu slúži užívateľsky definovaná funkcia, ktorej vstupom sú:

- obsah zachyteného paketu včetne hlavičiek sieťových protokolov
- odkaz na inštanciu štruktúry kľúčových polí definovanú užívateľom (viď. podkapitola 3.2.1)
- odkaz na inštanciu štruktúry atribútov definovanú užívateľom (viď. podkapitola 3.2.1)

Vo funkcií užívateľ podľa vlastných požiadaviek priradí položkám štruktúr hodnoty, ktoré získa extrakciou dát z obsahu paketu. Výstupom funkcie je štruktúra naplnená kľúčovými poľami paketu resp. toku (na základe ktorej je následne nájdený tok, ktorému paket náleží) a štruktúra naplnená atribútami toku (ktorými sú neskôr aktualizované príslušné dátové položky toku).

Návratovou hodnotou funkcie užívateľ špecifikuje, či má byť daný paket spracovaný (tzn. vyhľadanie a aktualizácia príslušného toku) alebo zahodený. Týmto spôsobom môže užívateľ implementovať filter na zachytávanú sieťovú premávku.

3.2.3 Aktualizácia záznamu toku

Ďalšou akciou, ktorá je v réžií užívateľa je aktualizácia záznamu toku dátami extrahovanými z paketu. K tomuto účelu slúži v poradí druhá užívateľom definovaná funkcia. Jej vstupmi sú:

- odkaz na inštanciu štruktúry atribútov naplnenú v užívateľskej funkcií pre extrakciu dát z paketu

- odkaz na štruktúru atribútov v zázname toku, ktorému paket náleží
- pravdivostná hodnota udávajúca či sa jedná o prvý paket toku alebo nie

Úlohou užívateľa je aktualizovať hodnoty atribútov v zázname toku podľa jeho požiadaviek. K dispozícií má atribúty získané z aktuálneho paketu a príznak, ktorý udáva, či sa jedná o prvý paket toku – v takom prípade môže byť totiž aktualizácia záznamu odlišná (napr. môže užívateľ chcieť nastaviť atribút časové razítko začiatku toku, ktorý sa už neskôr nemení). K uvedeným informáciám má užívateľ pre aktualizáciu záznamu naviac k dispozícií nasledovné informácie zo sondy:

- aktuálna vzorkovacia frekvencia
- hodnota aktívneho timeoutu
- hodnota neaktívneho timeoutu
- časové razítko momentu spustenia sondy
- počet obdržaných, spracovaných a zahodených paketov
- počet IPFIX správ odoslaných sondou kolektoru a ich veľkosť v bajtoch
- celkový počet záznamov tokov odoslaných v IPFIX správach

3.3 Rozhranie pre prístup k sonde

Pre prístup k sonde počas jej behu bol navrhnutý obslužný program s textovým rozhraním. Výhodou textového rozhrania je, že nepotrebuje k svojmu spusteniu operačný systém s grafickým rozhraním a operátor tak môže pristupovať k sonde z ľubovolného terminálu pripojeného k sieti. Rozhranie poskytuje užívateľovi nasledujúce možnosti:

- zobraziť štatistiky merania (počet spracovaných a zahodených paketov, priemerná rýchlosť spracovania dát, čas spustenia sondy, štatistiky exportu, stav pamäte sondy apod.)
- zobraziť aktuálne nastavenia sondy
- nastaviť vzorkovaciu frekvenciu sondy
- nastaviť novú hodnotu aktívneho timeoutu
- nastaviť novú hodnotu neaktívneho timeoutu
- zapísať aktuálny obsah pamäte sondy do súboru

Komunikácia so sondou je realizovaná prostredníctvom sieťových soketov. Sonda vystupuje v komunikácií v roli serveru a obslužný program v roli klienta. Vzhľadom k definovanému formátu komunikácie formou jednoduchého sieťového protokolu je možné ľahko implementovať obslužný program sondy s ľubovolným rozhraním a vstupom/výstupom. Na prenos správ je použitý transportný protokol SCTP.

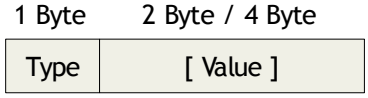
3.3.1 Administračný server

Administračný server, ktorý je súčasťou sondy je navrhnutý ako konkurentný blokujúci sieťový server. Čaká sa spojenie od obslužného programu na zvolenom SCTP porte.

Beh serveru je umiestnený do samostatného vlákna, takže obsluha klienta neprerušuje meranie prevádzané sondou. Výnimkou je prípad, keď server obdrží požiadavku na zaslanie aktuálneho obsahu pamäte sondy. V tom prípade je kvôli integrite dát pamäť sondy zamknutá až do kompletného odoslania jej obsahu obslužnému programu, takže merací proces sondy do nej nemôže v tom čase zapisovať. To znamená pri predpokladanej veľkosti jedného záznamu toku 36 bajtov a počte 100 000 záznamov v pamäti sondy naformátovanie a prenesenie cca 3,4 MiB dát. Merací proces sondy bude v takom prípade vyradený z činnosti niekoľko jednotiek až desiatok sekúnd, v závislosti od rýchlosti prenosu dát medzi sondou a obslužným programom.

3.3.2 Komunikačný protokol

Komunikácia medzi sondou a obslužným programom prebieha pomocou jednoduchého komunikačného protokolu obsahujúceho 2 správy – požiadavka a odpoveď. Požiadavky sú zasielané výhradne obslužným programom sonde a odpovede výhradne sondou obslužnému programu. Formát správy požiadavky je na obrázku 3.4.



Obrázok 3.4: Formát požiadavky

Existuje 6 typov požiadavkov:

- `Type = 1` – žiadosť o zaslanie štatistík merania
- `Type = 2` – žiadosť o zaslanie nastavení sondy
- `Type = 3, Value = 32-bitové celé číslo` – žiadosť o nastavenie novej hodnoty vzorkovacej frekvencie na hodnotu `Value`
- `Type = 4, Value = 16-bitové celé číslo` – žiadosť o nastavenie novej hodnoty aktívneho timeoutu na hodnotu `Value`
- `Type = 5, Value = 16-bitové celé číslo` – žiadosť o nastavenie novej hodnoty neaktívneho timeoutu na hodnotu `Value`
- `Type = 6` – žiadosť o zaslanie celého obsahu pamäte sondy

Formát odpovede sa líši v závislosti na type požiadavku. Pri požiadavkoch typu 1 a 2 sú v odpovedi zaslané hodnoty čítačov jednotlivých štatistík resp. hodnoty nastavení sondy. V prípade požiadavkov typu 3, 4 a 5 ide o textové reťazce informujúce o úspešnosti/neúspešnosti zadaných akcií. U typu 6 je v prvej správe zaslaná 32-bitová hodnota počítadla záznamov, ktorá udáva koľko správ, obsahujúcich naformátované záznamy tokov, bude následne ešte zaslaných.

3.3.3 Formát výstupu pamäte sondy

Pre zápis pamäte sondy do súboru bol zvolený formát CSV [25]. Ide o jednoduchý textový formát pre ukladanie dát do súborov. Každý záznam je uložený na samostatný riadok a ako oddeľovač jednotlivých polí záznamu je použitý znak čiarka. Jeden záznam v našom prípade reprezentuje záznam jedného sieťového toku a polia záznamu reprezentujú jednotlivé atribúty toku. Príklad záznamu toku, ktorého formát je definovaný v podkapitole 3.2.1 vyzerá nasledovne (v poradí – identifikátor protokolu, zdrojová IPv4 adresa, cieľová IPv4 adresa, zdrojový port, cieľový port, časové razítko začiatku toku, časové razítko ukončenia toku, počet bajtov toku):

```
17,147.229.196.252,91.213.160.118,2990,80,1274621830,1274621847,82050
```

V prípade, že hodnota niektorého z polí obsahuje znak čiarka alebo znak nového riadku musí byť táto hodnota uzavretá do úvodzoviek.

Výhodou tohto formátu je, že je ľahko čitateľný pomocou bežného textového editoru a je pomerne rozšírený – dokáže ho importovať väčšina databázových programov a tabuľkových procesorov.

4 Implementácia

Cieľovou platformou pre implementáciu sondy je operačný systém Linux. Jednou z dôležitých požiadaviek na sondu je rýchle spracovanie dát a s tým súvisiaci efektívny prístup do pamäte. Pre implementáciu je preto vhodnejší kompilovaný, nízkoúrovňový jazyk. Na druhú stranu ide o rozsiahlejší projekt, ktorý by s výhodou využil objektový návrh. Z týchto dôvodov bol pre implementáciu zvolený jazyk C++, ktorý poskytuje pokročilé rysy vyšších programovacích jazykov (OOP, polymorfizmus, generické programovanie) ako aj možnosti nízkoúrovňových jazykov (priamy prístup do pamäte, statická typová kontrola).

4.1 Použité nástroje

Pri vývoji sondy bol použitý prekladač `g++` (GNU C++) verzie 4.4.1. Norma IPFIX vyžaduje pre odosielanie záznamov tokov použitie transportného protokolu SCTP, pre jeho implementáciu sme použili knižnicu `libsctp`, ktorá rozširuje Berkley sokety o podporu tohto protokolu. Ďalšou knižnicou použitou pri vývoji je `PF_RING`. Tá implementuje sieťový soket pre efektívne zachytávanie paketov, bez generovania prerušení jadra operačného systému. Pre správu vlákien a časovačov sú použité knižnice z balíka `Boost`, ktoré sú voľne dostupné aj pre komerčné použitie. Knižnica implementujúca `PF_RING` je dostupná pod licenciou GPL.

Pri testovaní výkonnosti boli použité voľne dostupné programy `netperf` (pre testovanie prenosových rýchlostí sietí) a `tcpreplay` (pre generovanie sieťovej premávky opakovaným zasielaním premávky uloženej v `pcap` súboroch).

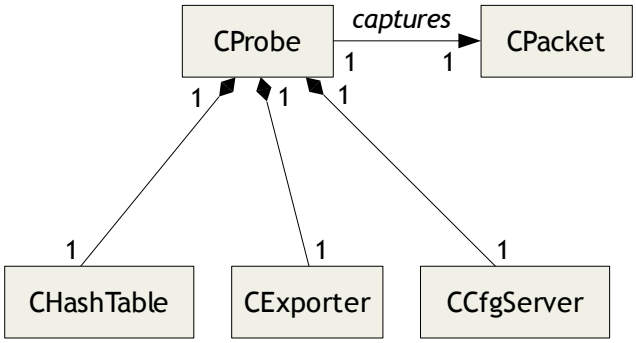
4.2 Štruktúra aplikácie

Implementácia sondy je postavená na objektovom návrhu. Jednotlivé logické časti sondy sú zapúzdrené do tried, čo výrazne zvyšuje prehladnosť štruktúry výsledného zdrojového kódu. Diagram tried aplikácie je na obrázku 4.1.

CProbe

Trieda `CProbe` je základnou triedou aplikácie, ktorá riadi celý proces behu sondy. Pri vytvorení inštancuje objekty tried `CHashTable`, `CExporter` a `CCfgServer` a v oddelenom vlákne spustí časovač, ktorý riadi proces údržby tokov. `CProbe` obsahuje implementáciu zachytávania paketov na zadanom sieťovom rozhraní. Po obdržaní paketu inicializuje objekt triedy `CPacket` a ten odovzdá na ďalšie spracovanie triede `CHashTable`. Obsahuje tiež čítače ukladajúce štatistiky priebehu merania

a zabezpečuje ich aktualizáciu prípadne prístup k nim ďalším triedam, ktoré ich potrebujú aktualizovať.



Obrázok 4.1: Štruktúra sondy – diagram tried

CPacket

Trieda `CPacket` je pri zachytení nového paketu inicializovaná jeho obsahom. Volaním užívateľom definovanej funkcie `usr_parse_pkt` extrahuje z paketu kľúčové polia a ďalšie atribúty paketu, ktoré potrebujeme v súvislosti s aktualizovaním záznamu príslušného toku. Z kľúčových polí toku vypočíta haš, ktorý bude následne použitý pre vyhľadanie záznamu toku.

CHashTable

Táto trieda obsahuje implementáciu pamäti záznamov tokov ako je popísaná v podkapitole 3.1.1. Volaním členskej metódy `update_flow(CPacket & Packet)` je jej predaný aktuálne spracovávaný paket. Trieda sa pokúsi nájst a aktualizovať príslušný tok. V prípade, že vyhľadávanie skončí neúspešne prebehne vytvorenie a vloženie nového záznamu toku. K tomuto účelu trieda implementuje Cuckoo hašovanie, popísané v 3.1.3.

CExporter

Trieda `CExporter` implemetuje exportér, ktorý sa stará o konštrukciu a odosielanie dátových záznamov a šablón vo formáte IPFIX. Jej činnosť je riadená časovačom zo samostatného vlákna aplikácie. Po obdržaní expirovaného záznamu toku vytvorí novú IPFIX správu a naformátuje do nej daný záznam. Aby nebol každý záznam zasielaný kolektoru v samostatnej IPFIX správe, čo by bolo dosť neefektívne, je v tomto momente spustený nový časovač s expiračnou dobou 1000 ms. Počas tejto doby môžu byť do vytvorenej IPFIX správy pridávané ďalšie záznamy. Správa je odoslaná na kolektor po uplynutí jej expiračnej doby alebo po jej naplnení, podľa toho, ktorá z udalostí nastane skôr. Maximálna veľkosť IPFIX správy je stanovená na 65 535 bajtov [1].

CCfgServer

CCfgServer obsahuje implementáciu administračného serveru sondy. Jej objekt je inštancovaný triedou CProbe tak, aby bežal v samostatnom vlákne. V cykle obsluhuje požiadavky zasielané obslužným programom (klientom).

4.3 Užívateľom definované štruktúry a funkcie

Pre činnosť sondy musí užívateľ špecifikovať kľúčové polia a ďalšie atribúty toku formou definície informačných elementov IPFIX ako položiek štruktúr. Ďalej musí uživateľ naplniť identifikátormi týchto infromačných elementov príslušné polia (postup popísaný v podkapitole 3.2.1). Tieto nastavenia sa vykonávajú v hlavičkovom súbore config.h.

Ďalšou povinnosťou užívateľa je definovať funkciu, ktorá naplní ním špecifikované atribúty toku dátami extrahovanými z paketu a funkciu, ktorá zabezpečí aktualizáciu záznamu toku príslušnými atribútmi. Definícia týchto 2 funkcií sa nachádza v súbore config.cpp. Deklarácia prvej menovanej funkcie vyzerá nasledovne:

```
bool usr_parse_pkt(const u_char *pkt,
                   struct pfring_pkthdr *hdr,
                   struct key_fields *keys,
                   struct data_fields *data);
```

Význam parametrov:

- pkt – pole bajtov obsahujúce obsah paketu vrátane hlavičky protokolu linkovej vrstvy
- hdr – ukazovateľ na štruktúru obsahujúcu niektoré extrahované dáta z obsahu paketu
- keys – ukazovateľ na štruktúru s kľúčovými položkami toku/paketu definovanými užívateľom
- data – ukazovateľ na štruktúru s ďalšími atríbútami toku/paketu definovanými užívateľom

Pre uľahčenie práce užívateľovi definícia funkcie v súbore config.cpp už obsahuje implementáciu extrakcie dát z hlavičiek protokolov IPv4, IPv6, TCP, UDP, SCTP, ICMP, ICMPv6 a IGMP. Je na užívateľovi aby pripravené dáta správne použil a za tým účelom sa musí oboznámiť so štruktúrami reprezentujúcimi hlavičky jednotlivých protokolov. Tie sú pre informáciu uvedené ako poznámka na konci súboru config.cpp. Deklarácia funkcie pre aktualizáciu záznamu toku, ktorému paket náleží vyzerá nasledovne:

```
void usr_update_flow(const struct data_fields *pkt_data,
                           struct data_fields *flow_data,
                     const struct internals *in,
                     bool first_pkt);
```

Význam parametrov:

- pkt_data – štruktúra obsahujúca atribúty práve spracovávaného paketu

- flow_data – štruktúra obsahujúca atribúty toku, ktorému daný paket náleží
- in – štruktúra obsahujúca aktuálne nastavenia a štatistiky sondy
- first_pkt – hodnota booleovského typu indikujúca, či sa jedná o prvý paket toku (a vytvárame teda nový záznam toku namiesto aktualizácie už existujúceho)

Úlohou užívateľa je v tejto funkcií aktualizovať atribúty záznamu toku, na ktoré ukazuje premenná flow_data hodnotami, ktoré získal extrakciou dát z paketu vo funkcií usr_parse_pkt() (na tie ukazuje premenná pkt_data).

4.4 Rozhranie obslužného programu

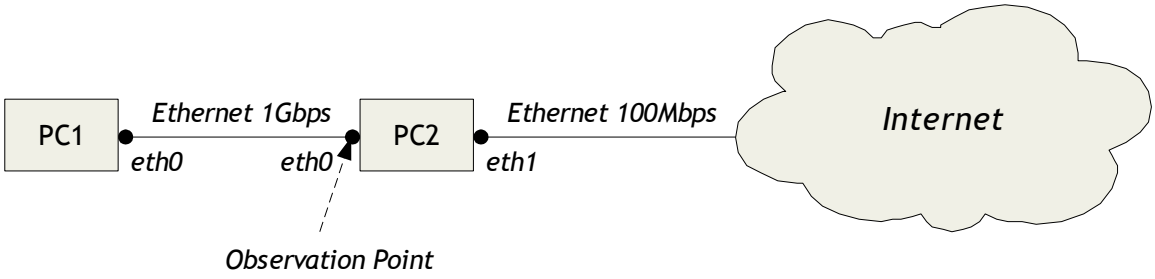
```
root@b05-638a:~# probe_ctl
Available commands: [1] print probe's statics   [2] print probe's settings      [3] set sampling rate
                    [4] set active timeout      [5] set inactive timeout       [6] dump probe's cache to file
                    [7] exit
Enter your choice: 1
-- Connected to 127.0.0.1:52010

--------------------- Measuring statics ---------------------
Number of packets received:       143287
Number of packets processed:      143287
Number of packets dropped:        0
Number of packets buffered:       0
% of packets dropped:             0.00

Size of processed packets:        14013921 bytes
Average data rate:                1.437 Mbit/s
Average packet processing rate:   1837.01 pkts/s
Measuring started at:             Tue May 25 22:01:54 2010

Number of active flows in cache:  114
Size of flow cache:               1048576

--------------------- Exporting statics ---------------------
Number of IPFIX messages send to collector:     0
Number of flows exported to collector:          0
Size of one flow record:                        0 bytes
Total size of messages sent to collector:       0 bytes


Available commands: [1] print probe's statics   [2] print probe's settings      [3] set sampling rate
                    [4] set active timeout      [5] set inactive timeout       [6] dump probe's cache to file
                    [7] exit
Enter your choice:
```

Na obrázku 4.2 sa nachádza ukážka implementovaného textového rozhrania obslužného programu sondy.

Obrázok 4.2: Rozhranie obslužného programu sondy

5 Experimenty a merania

Pre testovanie a meranie výkonnosti sondy bola použitá jednoduchá sieť zložená z dvoch osobných počítačov zapojených podľa schémy na obrázku 5.1. Priepustnosť spojenia medzi počítačmi PC1 a PC2 bola zistená nástrojom netperf a priemerne dosahovala hranicu 630 Mbps v oboch smeroch.



Obrázok 5.1: Testovacia konfigurácia

Konfigurácia počítača PC1:

- procesor: Intel Pentium 4 Northwood 2,26 GHz
- pamäť: 1 GB DDR 266
- sieťová karta: Realtek 8169 (Ethernet 1 Gbps)

Konfigurácia počítača PC2:

- procesor: Intel Pentium M 1,7 GHz
- pamäť: 1 GB DDR 400
- sieťová karta: Intel PRO/1000 MT (Ethernet 1 Gbps)

5.1 Metodika

Metodika merania spočívala v generovaní sieťovej premávky počítačom PC1 a jej zasielanie rozhraním eth0 na počítač PC2. Na rozhraní eth0 počítača PC2 bola spustená sonda, ktorá merala priebeh premávky zasielanej počítačom PC1. Pre generovanie premávky slúžil nástroj tcpreplay, ktorý dokáže znovu posielať zachytenú sieťovú premávku uloženú vo formáte pcap. Pre všetky uvádzané testy bola použitá rovnaká vzorka reálnej premávky nasledovných parametrov:

- dĺžka trvania premávky – 900 sekúnd
- počet paketov – 12 049 730

- celková velkosť paketov – 7182 MB
- priemerný dátový tok – 67 Mbps
- celkový počet IPv4 adries – 527 851
- celkový počet IPv6 adries – 885
- zloženie premávky – 90% TCP, 8% UDP, 1% ICMP, 1% ostatné

Tento vzor sieťovej premávky bol posielaný nástrojom tcpreplay s voľbou -t, ktorá zabezpečí, že dáta nie sú posielané originálnou rýchlosťou, ale maximálnou rýchlosťou akú je schopný generujúci systém dosiahnuť. Týmto spôsobom odoslal počítač PC2 celý vzor premávky priemerne za 192 sekúnd a generoval tak priemerný tok ~ 62 500 paketov za sekundu.

Pre meranie boli použité tri rôzne konfigurácie pre záznam toku, každá s rôznym počtom atribútov a tým aj náročnosťou spracovania. Prehľad konfigurácií poskytuje tabuľka 5.1. Veľkosť pamäte pre záznamy tokov v sonde bola počas testov nastavená na hodnotu 1 048 576 položiek.

	konfiguácia 1	konfiguácia 2	konfiguácia 3
kľúčové polia	sourceIPv4Address destinationIPv4Address	protocolIdentifier sourceIPv4Address destinationIPv4Address sourceTransportPort destinationTransportPort	protocolIdentifier sourceIPv4Address destinationIPv4Address sourceTransportPort destinationTransportPort
atribúty	octetDeltaCount	flowStartSeconds flowEndSeconds octetDeltaCount	flowStartSeconds flowEndSeconds octetDeltaCount packetDeltaCount minimumIpTotalLength maximumIpTotalLength exportedMessageTotalCount droppedPacketDeltaCount octetDeltaSumOfSquares
veľkosť záznamu	16 bajtov	29 bajtov	77 bajtov

Tabuľka 5.1: Testovacie konfigurácie záznamu toku

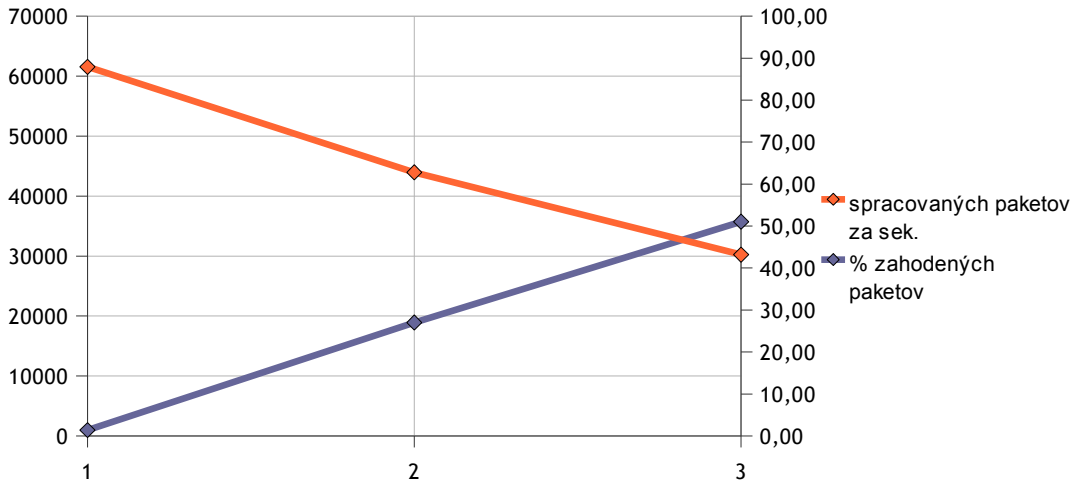
Každá z konfigurácií je špecifická počtom atribútov uchovávaných pre záznam toku a tým veľkosťou tohto záznamu. To má dopad na objem spracovávaných dát a rovnako na objem dát odosielaných kolektoru.

5.2 Výkonnosť v závislosti od spôsobu výstupu

Pri obvyklom režime práce sondy sú expirované záznamy tokov odosielané kolektoru. Sondu je však možné spustit aj v režime, keď záznamy tokov vypisuje v textovej podobe na štandardný výstup, prípadne záznamy tokov neposiela na žiadny výstup. Pri meraní bola použitá konfigurácia č. 2 záznamu toku, uvedená v tabuľke 5.1. Dĺžka záznamu jedného toku je 29 bajtov. Tabuľka 5.2. zhrňuje výkonnosť sondy v týchto troch režimoch. Namerané dáta sú vynesené v grafe na obrázku 5.2.

	bez exportu dát	export na kolektor	textový výstup
počet prijatých paketov	12049730	12049730	12049730
počet spracovaných paketov	11880469	8794809	5896746
počet zahodených paketov	169261	3254931	6152984
% zahodených paketov	1,40	27,01	51,06
spracovaných paketov za sek.	61557	43974	30240

Tabuľka 5.2: Závislosť výkonnosti sondy od spôsobu výstupu



Obrázok 5.2: Závislosť výkonnosti sondy od spôsobu výstupu

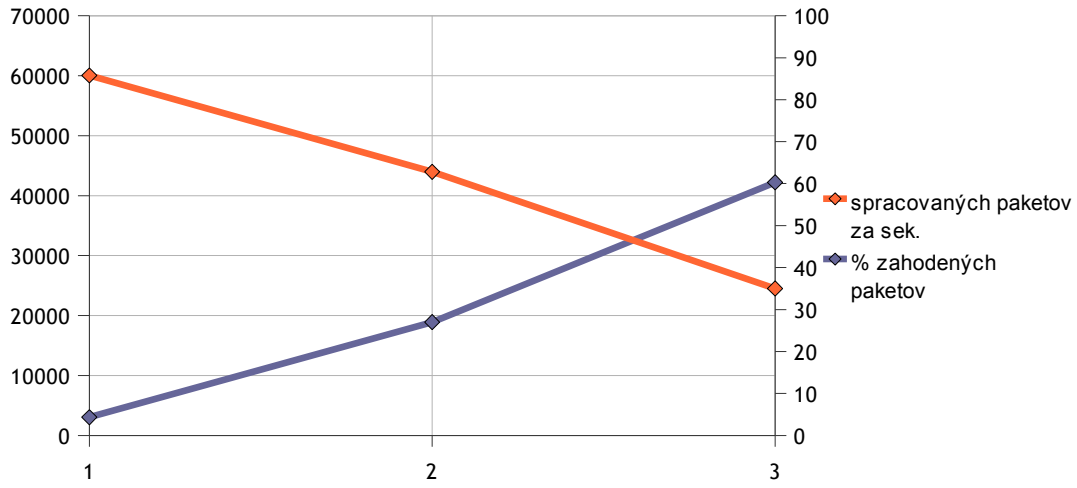
Zo získaných hodnôt je vidieť, že úzkym hrdlom výkonnosti sondy sú prestoje pri exporte záznamov. Bolo by výhodné exportér vybaviť vyrovnávacou pamäťou, do ktorej by merací proces mohol zapisovať nové dáta aj v prípade, že odosielanie predchádzajúcich ešte nie je dokončené.

5.3 Výkonnosť v závislosti od formátu záznamu

Dĺžka záznamu toku priamo ovplyvňuje objem exportovaných dát, takže na základe výsledkov meraní v predchádzajúcej kapitole môžme usudzovať, že výkonnosť sondy bude klesať priamo úmerne so zvyšujúcou sa dĺžkou záznamu. Pri meraní boli použité 3 rôzne konfigurácie záznamov toku uvedené v tabuľke 5.1 a záznamy boli exportované na kolektor bežiaci na rovnakom počítači ako sonda. Namerané dáta sú uvedené v tabuľke 5.3 a závislosti vynesené v grafe na obrázku 5.3.

	formát záznamu toku		
	konfigurácia 1	konfigurácia 2	konfigurácia 3
počet prijatých paketov	12049737	12049730	12049730
počet spracovaných paketov	11528043	8794809	4780876
počet zahodených paketov	521694	3254931	7268854
% zahodených paketov	4,33	27,01	60,32
spracovaných paketov za sek.	60042	43974	24517

Tabuľka 5.3: Závislosť výkonnosti sondy od formátu záznamu toku



Obrázok 5.3: Závislosť výkonnosti sondy od formátu záznamu toku

Z výsledkov meraní môžeme vypočítať teoretické hranice priepustnosti sondy. Pri zázname toku o dĺžke 16 bajtov dokáže sonda spracovávať prichádzajúce pakety rýchlosťou 60 000 paketov za sekundu. To dáva pri veľkosti paketov 64 bajtov teoretickú priepustnosť 29 Mbps. Pri veľkosti paketov 1500 bajtov je teoretická priepustnosť sondy 686 Mbps.

Pri zázname toku o dĺžke 77 bajtov dokáže sonda spracovať 24 500 paketov za sekundu. Priepustnosť sa v tomto prípade pohybuje od 12 Mbps (pri spracovávaných paketoch o veľkosti 64 bajtov) do 280 Mbps (pri spracovávaných paketoch o veľkosti 1500 bajtov).

6 Záver

Prvá časť práce obsahuje prehľad problematiky merania sietí a oboznamuje čitateľa so zavedenou terminológiou v tomto odvetví. Detailnejšie je popísané meranie na základe dátových tokov, ktoré je dnes jedným z najpoužívanejších prístupov k meraniu sieťovej premávky.

V ďalšej časti je uvedený teoretický rozbor problematiky a sú detailnejšie popísané techniky, princípy a algoritmy, ktoré sú v súvislosti s meraním tokov bežne zavedené, ako aj nové prístupy prezentované na odborných konferenciách v posledných rokoch. Detailnejší je popis tých princípov, ktoré budú použité pri implementácií sondy, ktorej návrh je predmetom tejto diplomovej práce. V tomto smere poskytuje druhá kapitola teoretické východiská pre riešenie daného problému. Značnú časť druhej kapitoly práce pokrýva prehľad protokolu IPFIX používaného pre export záznamov nameraných tokov.

Najrozsiahlejšou časťou práce je kapitola zaoberajúca sa návrhom architektúry sondy pre meranie tokov na sieti. Dôraz bol kladený na flexibilitu záznamu, pretože medzi voľne šíriteľnými nástrojmi nie je dostupné riešenie, ktoré by poskytovalo užívateľovi ozajstnú flexibilitu. U sondy, ktorá je výstupom tejto práce má užívateľ možnosť definovať si a používať aj doposiaľ neimplementované atribúry pre záznam toku. Sonda používa efektívnu techniku indexovania záznamov, veľkosť pamäte je nastaviteľná a užívateľ má tak možnosť regulovať pamäťové nároky sondy.

V závere práce sú uvedené výsledky niektorých experimentov a testov, ktoré boli na implementovanej sonde vykonané. Z nich vyplýva, že úzkym hrdlom sondy sú momentálne prestoje pri prenose dát z exportéru do kolektoru, kedy merací proces nemôže zadávať exportéru na odoslanie dáta, kým exportér nedokončí odosielanie predošlých expirovaných záznamov. Táto nevýhoda by sa dala odstrániť implementovaním vyrovnávacej pamäte v exportéri. Táto ulóha je jedným z námetov na pokračovanie tohto projektu.

Literatúra

[1] Claise, B.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101, Internet Engineering Task Force, Január 2008.

[2] Quittek, J., Zseby, T., Claise, B., Zander, S.: Requirements for IP flow Information Export (IPFIX). RFC 3917, Internet Engineering Task Force, Október 2004.

[3] Quittek, J., Bryant, S., Claise, B., Aitken, P., Meyer, J.: Information Model for IP Flow Information Export. RFC 5102, Internet Engineering Task Force, Január 2008.

[4] Zseby, T., Molina, M., Duffield, N., Niccolini, S., Raspall, F.: Sampling and Filtering Techniques for IP Packet Selection. RFC 5475, Internet Engineering Task Force, Marec 2009.

[5] Claise, B., Johnson, A., Quittek, J.: Packet Sampling (PSAMP) Protocol Specifications. RFC 5476, Internet Engineering Task Force, Marec 2009.

[6] Estan, C., Keys, K., Moore, D., Varghese, G.: Building a better netflow. Proceedings ACM SIGCOMM, 2004.

[7] Cisco IOS NetFlow. Dostupné na URL (december 2009): http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html

[8] Internet usage statistics. Dostupné na URL (december 2009): http://www.internetworldstats.com/stats.htm

[9] Soľanka, L.: Návrh architektúry sondy pro monitorování síťových toků [Diplomová práce]. Vysoké učení technické, Fakulta informačních technilogií, 2009.

[10] Špringl, P.: Architektura programového vybavení monitorovací sondy na bázi toků [Diplomová práce]. Vysoké učení technické, Fakulta informačních technilogií, 2009.

[11] Welte, H.: Flow-based network accounting with Linux. Proceedings of the Linux Symposium, Ottawa, Ontario, Canada, Júl 2005. 265-270 s.

[12] Garcia, L.: Programming with Libpcap – Sniffing the Network From Our Own Application. Haking 2/2008, 38-46 s. ISSN 1733-7186.

[13] Deri, L.: Improving Passive Packet Capture: Beyond Device Polling. Pisa, Italy. Dostupné na URL (január 2010): http://luca.ntop.org/Ring.pdf

[14] Deri, L.: Why TNAPI (Threaded NAPI)? Dostupné na URL (január 2010): http://www.ntop.org/TNAPI.html

[16] Pagh, R., Rodler, F.: Cuckoo Hashing. Dánsko, 2001. Dostupné na URL (január 2010): http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.25.4189&rep=rep1&type=pdf

[17] Erlingsson, U., Manasse, M., Mcsherry, F.: A cool and practical alternative to traditional hash tables. California, USA, 2006. Dostupné na URL (január 2010): http://www.ru.is/faculty/ulfar/CuckooHash.pdf

[18] Zukowski, M., Héman, S., Boncz, P.: Architecture-Conscious Hashing. Proceedings of the International Workshop on Data Management on New Hardware (DaMoN), Jún 2006.

[19] Ross, K.: Efficient Hash Probes on Modern Processors. IBM Research Report RC24100, 8 november 2006. Dostupné na URL (január 2010): http://domino.research.ibm.com/library/cyberdig.nsf/papers/DF54E3545C82E8A585257222006FD9A2/$File/rc24100.pdf

[20] Dressler, F., Münz, G.: Flexible Flow Aggregation for Adaptive Network Monitoring. Proceedings of 31st IEEE Conference on Local Computer Networks (LCN): 1st IEEE LCN Workshop on Network Measurements, Tampa, FL, November 2006. 702-709 s.

[21] Hu, Y., Chiu, D.M., Lui, J.: Adaptive Flow Aggregation - A New Solution for Robust Flow Monitoring under Security Attacks. In IEEE/IFIP Network Operations and Management Symposium (IEEE/IFIP NOMS 2006). 2006, 424-435 s.

[22] Jenkins, B.: A Hash Function for Hash Table Lookup. Dostupné na URL (máj 2010): http://www.burtleburtle.net/bob/hash/doobs.html

[23] Hash functions: An empirical comparison. Dostupné na URL (máj 2010): http://www.strchr.com/hash_functions

[24] IP Flow Information Export (IPFIX) Information Elements. IANA, máj 2007. Dostupné na URL (máj 2010): http://www.iana.org/assignments/ipfix/ipfix.xhtml

[25] Shafranovich, Y.: Common Format and MIME Type for Comma-Separated Values (CSV) Files. RFC 4180, Internet Engineering Task Force, Október 2005.

Zoznam príloh

Príloha A. Inštalačná a užívateľská príručka
Príloha B. Obsah priloženého dátového nosiča
Príloha C. Dátový nosič CD s implementáciou, potrebnými knižnicami a zdrojovým textom práce

Príloha A

Inštalačná a užívateľská príručka

Priložené CD obsahuje zdrojové súbory sondy a zdrojové súbory knižníc Boost a PF_RING. Obsahom nosiča je tiež implementácia triviálneho SCTP serveru, ktorý zastupuje úlohu kolektoru, keďže momentálne neexistuje žiadna voľne dostupná implementácia IPFIX kolektoru podporujúceho protokol SCTP. Tento server prijíma IPFIX správy od sondy a dokáže ich obsah zobrazovať na štandardný výstup.

K prekladu sondy je nutná podpora protokolu SCTP v jadre operačného systému a hlavičkové súbory knižnice libsctp (príslušný balík má názov libsctp-dev v distribúciách Ubuntu/Debian a libsctp-devel v distribúciách Fedora/RedHat). Väčšina aktuálnych Linuxových distribúcií je dodávaná s podporou protokolu SCTP priamo v distribučnom jadre.

Inštalácia knižnice PF_RING:

```
$ cd PF_RING/kernel/
$ make
$ make install
$ modprobe pf_ring
$ cd ../userland/lib/
$ make
$ make install
```

Inštalácia knižníc Boost:

```
$ cd boost/
$ tar xzf boost_1_42_0.tar.gz
$ cd boost_1_42_0/
$ ./bootstrap.sh --prefix=/usr/local \
  --with-libraries=system,thread,date_time
$ ./bjam install
```

Inštalácia sondy zo zdrojových kódov:

```
$ cd ipfix_probe/
$ make
$ make install
```

Nastavenie formátu záznamu toku:

Formát záznamu toku špecifikuje užívateľ definíciou informačných elementov IPFIX formou položiek štruktúr key_fields a data_fields v hlavičkovom súbore config.h. Štruktúra key_fields

obsahuje definíciu kľúčových polí toku a štruktúra data_fields definíciu sledovaných atribútov toku. Príklad konfigurácie:

```
struct key_fields {
    unsigned8   protocolIdentifier;
    ipv4Address sourceIPv4Address;
    ipv4Address destinationIPv4Address;
    unsigned16  sourceTransportPort;
    unsigned16  destinationTransportPort;
}

struct data_fields {
    dateTimeSeconds flowStartSeconds;
    dateTimeSeconds flowEndSeconds;
    unsigned64      octetDeltaCount;
}
```

Zoznam dostupných informačných elementov a im prislúchajúcich typov je možné nájsť napríklad na http://www.iana.org/assignments/ipfix/ipfix.xhtml alebo v RFC 5102. Užívateľ musí v hlavičkovom súbore config.h taktiež inicializovať príslušné polia identifikátormi zadaných informačných elementov s predponou e_. Príklad:

```
u_int16_t key_fields_array[] = {e_protocolIdentifier,
                                e_sourceIPv4Address,
                                e_destinationIPv4Address,
                                e_sourceTransportPort,
                                e_destinationTransportPort};

u_int16_t data_fields_array[] = {e_flowStartSeconds,
                                 e_flowEndSeconds,
                                 e_octetDeltaCount};
```

V súbore config.cpp musí užívateľ definovať priradenie dát extrahovaných z paketu jednotlivým definovaným IPFIX elementom vo funkcií usr_parse_pkt() a aktualizáciu atribútov toku vo funkcií usr_update_flow().

Spustenie sondy:

```
$ ipfix_probe [-i interface] [-l administration_port] [-c collector_ip]
              [-p collector_port] [-s sampling_rate] [-a active_timeout]
              [-t inactive_timeout] [-h] [-v]
```

```
VOĽBY:
  -i interface
     Špecifikuje sieťové rozhranie na ktorom bude sonda zachytávať pakety.
     Ak voľba nie je zadaná sonda zachytáva pakety na rozhraní 'eth0'.

  -l administration_port
     Špecifikuje SCTP port, na ktorom bude počúvať administračný server
     sondy. Ak voľba nie je zadaná použije sa predvolená hodnota 52010.

  -c collector_ip
     Špecifikuje IP adresu kolektoru, na ktorý sa majú posielať záznamy
     tokov. Ak voľba nie je zadaná záznamy tokov sa neodosielajú na
     kolektor.

  -p collector_port
     Špecifikuje SCTP port kolektoru, na ktorý sa majú posielať záznamy
     tokov. Ak voľba nie je zadaná použije sa predvolená hodnota 4739.

  -s sampling_rate
     Špecifikuje vzorkovaciu frekvenciu pre spracovanie prichádzajúcich
     paketov. Ak voľba nie je zadaná spracovávajú sa pakety bez
     vzorkovania.

  -a active_timeout
     Špecifikuje hodnotu aktívneho timeoutu v sekundách. Ak voľba nie je
     zadaná použije sa predvolená hodnota 900.

  -t inactive_timeout
     Špecifikuje hodnotu neaktívneho timeoutu v sekundách. Ak voľba nie
     je zadaná použije sa predvolená hodnota 20.

  -o
     Nastaví výpis expirovaných tokov na štandardný výstup.

  -h
     vypíše zoznam volieb programu

  -v
     vypíše verziu programu
```

Spustenie obslužného programu sondy:

```
$ probe_ctl [-c ip_address] [-p port] [-1] [-2] [-h] [-v]
```

```
VOĽBY:
  -c ip
     Špecifikuje IP adresu zariadenia, na ktorom je spustená sonda, ku
     ktorej sa chce operátor pripojiť. Ak voľba nie je zadaná obslužný
     program sa pokúsi nadviazať spojenie na localhost.

  -p collector_port
     Špecifikuje port na ktorom počúva administračný server sondy. Ak
     voľba nie je zadaná obslužný program sa pokúsi nadviazať spojenie na
     predvolený port 52010.
  -1
     Voľba zamedzí spusteniu programu v interaktívnom móde. Pripojí sa
     k sonde, zistí a vypíse štatistiky merania a skončí.

  -2
     Voľba zamedzí spusteniu programu v interaktívnom móde. Pripojí sa
     k sonde, zistí a vypíse aktuálne nastavenia sondy a skončí.

  -h
     vypíše zoznam volieb programu

  -v
     vypíše verziu programu
```

Spustenie SCTP serveru zastupujúceho kolektor:

```
$ ipfix_probe/collector/collector [-p]

VOĽBY:
  -p
     ak je voľba zadaná, server vypisuje obsah prijatých IPFIX správ po
     bajtoch na štandardný výstup
```

Príloha B

Obsah priloženého dátového nosiča

- ipfix_probe – obsahuje zdrojové súbory sondy
- ipfix_probe/collector – obsahuje zdrojové kódy triviálneho testovacieho kolektoru
- boost – obsahuje zdrojové súbory knižnice Boost
- PF_RING – obsahuje zdrojové súbory knižnice PF_RING
- dp.odt – zdrojový tvar textu diplomovej práce
- dp.pdf – text diplomovej práce vo formáte pdf